UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Servidor para control y seguimiento de sensores y notificaciones en dispositivos móviles Android

Miguel Rojo Esteva
Tutor: Gonzalo Martínez Muñoz

Julio 2015

Resumen

El crecimiento desmesurado en los últimos años del uso de dispositivos móviles, ya sean smartphones o tablets (personales y de trabajo) que permiten a los usuarios estar conectados en todo momento a las redes sociales, mensajes instantáneos, correo electrónico, etc. está creando una dependencia cada vez mayor que nos obliga a llevar siempre con nosotros todos nuestros dispositivos.

Es un hecho que a día de hoy, ya sea por motivos laborales, de estudio o de ocio, cada vez pasamos más horas delante de un ordenador, con lo que parece evidente pensar que se puede mejorar la calidad de vida de los usuarios gracias a una de las herramientas más potentes, sino la más, accesible a toda la sociedad: la red. Esta es la premisa de la que nace este trabajo.

El proyecto que se presenta se basa en la creación de una aplicación web asociada a todos los dispositivos electrotécnicos de cada usuario que permitan la concentración de notificaciones recibidas en cada uno de ellos desde un único punto de acceso, entendiéndose como tales las notificaciones de mensajería instantánea, de eventos, de correos electrónicos, etc. así como su gestión y reenvío de forma sencilla y rápida. Además en este proyecto se ha desarrollado una aplicación móvil que servirá de intermediario entre la aplicación web y el dispositivo. Gracias a estas aplicaciones no será necesario tener cerca el móvil o la tablet, simplemente cada dispositivo debe tener instalada la App móvil desarrollada en este proyecto.

La aplicación web será una ventana a los contenidos más significativos de todos los dispositivos asociados a un usuario. La aplicación web permitirá ver: las notificaciones, sus ajustes, la geolocalización del dispositivo o la retransmisión, en tiempo real, de la captura de vídeo de la cámara del móvil. Esta última funcionalidad permite al usuario ver qué está pasando cerca de su dispositivo (se puede imaginar qué ventaja supondrían estos elementos en caso de robo, por ejemplo). Así mismo pueden realizarse otras acciones sobre los ajustes, como es la activación / desactivación del wifi o el bluetooth, alarmas, etc.

Es un proyecto cuyo fin es mejorar, en la medida de lo posible, la calidad de vida de los usuarios sin requerir tener alto conocimiento informático dada su simplicidad de manejo, agilidad y rapidez de adaptación. En su desarrollo se han utilizado herramientas punteras que dan óptimos resultados tanto en el flujo de información como en visualización web.

Palabras clave: notificación, streaming, toggle, Android, dispositivo, acceso.

Abstract

The enormous growth in recent years of the use of mobile services, whether smartphones or tablets (for personal and work tasks) allows users to be connected anytime to social networks, instant messaging, email, etc. This is producing a growing dependence to technologies that forces us to always carry with us all of our gadgets.

It is a real fact that today, whether for work, studies or leisure, we spend more hours in front of a computer, which makes a point for us to think that we can improve human life quality by one of the most powerful tools, but the most accessible one to all of society: the network.

This work is born because of the previous premise.

This project is based on the creation of a web application associated with all electrical devices of the user, which allows to concentrate the notifications received in each one of them from a single point of access, understanding them as instant messaging notifications, events, emails, etc., and their quickly and easily management and forward. Additionally, this project has developed a mobile application that acts as an intermediary between the web application and the device. Thanks to this application, it is not necessary to have near the phone or the tablet; all you need is just have downloaded the app.

The web application is a window to the most significant contents of the devices, notification, and their settings, from which you can see the geolocation or retransmission, in real time, of the captured video from the phone´s camera, which allows the user to see what is happening near his device (everyone can imagine the advantage these elements would involve in the event of theft, for example). Also, other actions on adjustments can be made, such as on/off wifi or Bluetooth, alarms, etc.

It is a project which aims to improve, as far as possible, the quality of life for users that do not require high computer knowledge because its simplicity of use, flexibility and speed of adaptation. In its development, leading tools have been used widely by developers, which ones work best both in the flow of information and web viewing.

Keywords: notification, streaming, toggle, Android, device, access.

Glosario

Android	Sistema operativo basado en el núcleo de Linux diseñado principalmente para dispositivos móviles táctiles.
SQL	Structured Query Language. Lenguaje estructurado de consulta que permite efectuar búsquedas con el fin de obtener la información alojada en la base de datos.
JavaScript	Lenguaje de programación interpretado empleado normalmente en el lado del cliente en los navegadores web.
GCM	*Google cloud messaging* servicio de google que permite enviar mensajes push a dispositivos móviles que tengan instalado Google Play Services.
AJAX	Asynchronous JavaScript and XML. Tecnología que permite la comunicación entre el servidor y el cliente de forma asíncrona.
jQuery	Librería JavaScript que permite manipular el código HTML de forma dinámica en tiempo de ejecución.
WebRTC	Web Real Time Communications. Es una funcionalidad agregada en los navegadores que permiten enviar audio y video en tiempo real.
HTML	Lenguaje de Marcado Hipertextual (del inglés HyperText Markup Language); lenguaje de marcas con las que se escriben las páginas Web.
CSS	Cascading Style Sheets. Lenguaje de formato que permite definir los estilos de una página web.
Smartphone	Dispositivo inteligente. En este documento son los dispositivos que tienen instalados el Sistema operative Android.
JSON	*JavaScript Object Notation,* formato para el intercambio de datos.
MVC	Modelo vista controlador.
NAT	Mecanismo empleado por routers IP para intercambiar paquetes entre dos redes.
STUN	Mecanismo que permite a clientes NAT encontrar su dirección IP pública.
TOGGLE	Son los botones que actúan como controles para activar o desactivar elementos (wifi, Bluetooth…)

Agradecimientos

A todos los profesores que a lo largo de la carrera me han proporcionado los conocimientos necesarios para que pueda realizar este proyecto.

A mi tutor especialmente por apoyarme, ayudarme y confiar en mí.

A mi familia por su apoyo incondicional y su paciencia.

A mis amigos que me han liberado de tensiones y me han hecho reír en los momentos en que el proyecto me generaba estrés. En especial a Jorge por ayudarme en tantos momentos en la carrera.

GRACIAS A TODOS

Índice

Índice de figuras

Índice de tablas

Estructura del documento

En este apartado se describen las actuaciones que describen las características para el desarrollo de cada uno de los bloques que componen la estructura de este proyecto.

Bloque I – Introducción. El desarrollo de este bloque tiene como objetivo presentar al lector los motivos que han dado lugar al proyecto objeto de este trabajo así como los objetivos, que constituyen los pilares del mismo.

Bloque II - Análisis. Se presenta la fase previa al desarrollo de la aplicación en la que se analiza la funcionalidad y los posibles casos de uso. Se presentan las distintas fases por las que pasa el proyecto desde su creación hasta la entrega al usuario final.

Bloque III – Tecnologías empleadas. En este bloque se describen, desarrollan y explican las tecnologías empleadas en el desarrollo de la aplicación, ya sean las tecnologías empleadas desde el lado del Smartphone como las empleadas desde el lado del cliente web.

Bloque IV - Arquitectura del Sistema. En este bloque se desarrollan los elementos de diseño tanto de la parte del servidor como del cliente web.

Bloque V – Diseño y desarrollo. Se estudian tres vías: análisis de tablas de la Base de Datos; análisis de diagramas de clases y descripción de cada uno de ellos en el desarrollo de la aplicación Android; y descripción de módulos en el desarrollo de la aplicación web.

Bloque VI - Pruebas. Se presentan diferentes pruebas de testeo del correcto funcionamiento de la aplicación.

Bloque VII - Conclusiones y Trabajo Futuro. Se describen las conclusiones obtenidas del desarrollo definitivo del proyecto. Se incluyen futuras mejoras que se implementarán en la aplicación.

Bloque I - Introducción

Motivación

En los últimos años se ha producido un crecimiento radical en el uso de dispositivos móviles inteligentes. Estos smartphones se han convertido en un elemento esencial en la vida de las personas, creando una dependencia de su uso para todo tipo de tareas.

Una de las características más importantes que tienen es su capacidad de mantenernos informados de todo tipo de contenido. Los desarrolladores de aplicaciones móviles tienen cada vez más presente esta virtud y cada vez son más las aplicaciones que nos notifican de eventos, información relevante, etc.

La sociedad actual tiene tal necesidad de tecnología móvil que han aparecido respuestas anómalas ante su ausencia. El hecho de que en un momento determinado puedas estar aislado de tu smartphone, o no disponer en tu trabajo de acceso a las notificaciones recibidas, origina un elevado nivel de estrés a un gran número de personas.

Actualmente, existe una demanda cada vez más importante de personas que exigen una mayor comodidad en el uso de la tecnología. Por otro lado ésta avanza a un ritmo superior al adquirido por los usuarios, por tanto hay que ofrecerles servicios de fácil aplicación y uso; crear una aplicación sencilla de utilizar para una mayoría.

Además, hoy día muchos usuarios necesitan llevar consigo constantemente varios dispositivos móviles, como pueden ser tablets y smartphones personales y de trabajo. Parece interesante el poder tener acceso a la información y notificaciones de todos estos dispositivos desde una pantalla sin tener que llevarlos necesariamente encima.

¿Qué ocurre en caso de pérdida o sustracción del dispositivo móvil? Por lo general se da por perdido. A todos nos gustaría poder localizarlo inmediatamente.

Así pues este proyecto surge como respuesta a una necesidad detectada en la sociedad actual, cada vez más necesitada de conexión a las redes y a la información, cada vez más atada a multitud de dispositivos móviles, cada vez más obligada por la propia sociedad y por los propios trabajos a esta dependencia.

Objetivos

El objetivo de este proyecto es la creación de una aplicación que aúne la información recibida por todos los dispositivos móviles disponibles de los usuarios en un único punto de acceso sin necesidad de llevarlos encima. Está dirigido principalmente a los usuarios que pasan gran cantidad de horas frente a un monitor, que en la actualidad, es un tanto por ciento elevadísimo de la población.

Desde este punto de acceso, vía web, se pueden realizar una serie de acciones sobre los dispositivos móviles asociados que se detallan a continuación:

1. **Acceso a los ajustes del Smartphone**. Esta plataforma permite modificar los ajustes de los dispositivos de forma remota, como son la activación / desactivación de la conexión wifi o del bluetooth. Pueden enviarse, además, vibraciones.

2. **Localización de la posición del móvil en tiempo real bajo demanda del usuario.** Haciendo uso del servicio GPS del dispositivo se puede localizar el teléfono móvil en cualquier momento. Esto ofrece grandes ventajas, ya sea por pérdida o por robo.

3. **Configuración de alarmas**. Desde la plataforma pueden configurarse las alarmas despertador, que se pueden emplear para realizar avisos.

4. **Recepción de notificaciones desde la web.** Todas las notificaciones que reciban los diferentes dispositivos, ya sea vía mensaje instantáneo, eventos de redes sociales, notificaciones de correos electrónicos, etc. son recibidos por medio del servidor en la página web de la aplicación en tiempo real.

5. **Envío de notificaciones desde la web a cualquiera de los dispositivos asociados**. Del mismo modo que en el punto anterior se explican las recepciones, desde la web pueden enviarse notificaciones a los dispositivos o reenviarlas de uno a otro.

6. **Envío de video en streaming desde la cámara del móvil a la web**. Desde la aplicación puedes acceder a la cámara de los dispositivos móviles y ver, en tiempo real lo capturado. Este punto, junto con el de geolocalización, permiten obtener mucha información en caso de robo para poder denunciar y, en el mejor de los casos, recuperar el dispositivo móvil. También puede permitir tener controlado algún elemento deseado simplemente colocando el dispositivo en ese lugar y utilizarlo como cámara de vigilancia.

7. **Internet de las cosas.** El concepto de que la red llegue a los objetos cotidianos es un hecho que se está produciendo a un ritmo frenético y eso se está logrando en gran medida gracias a Android. Con esta aplicación se busca tener una interconexión básica entre esos objetos.

Este proyecto tiene la capacidad de seguir aumentando sus prestaciones en función de las posibles demandas de los usuarios.

Bloque II - Análisis

Análisis de requisitos

El término análisis aplicado a sistemas significa descomponer el sistema en sus componentes para estudiar cada uno de ellos tanto como un ente aislado como en interacción con el resto de los componentes.

Cada uno de estos componentes constituye un requisito funcional, puesto que son necesarios para que se produzca el correcto funcionamiento de la aplicación.

A su vez existen los requisitos no funcionales, centrados en el rendimiento, que imponen restricciones en el diseño o la implementación aportando cualidades que debe cumplir el producto.

Requisitos funcionales

Recogen la funcionalidad que debe tener la aplicación, indicando qué cosas debe hacer. Se han agrupado los distintos requisitos en categorías.

Control de acceso

En este apartado se exponen los requisitos necesarios para el acceso a la aplicación y al cliente web.

Identificador	RF1.1
Nombre	Login
Descripción	Realice el login contra el servidor y compruebe que los datos introducidos sean correctos

Tabla 1: Requisito Funcional RF1.1

Identificador	RF1.2
Nombre	Registro
Descripción	Creación de un nuevo usuario e incluirlo a la base de datos

Tabla 2: Requisito Funcional RF1.2

Identificador	RF1.3
Nombre	Registro GCM
Descripción	Registro del dispositivo móvil frente al servidor de Google para la recepción de mensajes

Tabla 3: Requisito Funcional RF1.3

Filtrado

A continuación se muestran las funcionalidades relativas al filtrado de aplicaciones que representa cuales no deben enviar notificaciones al servidor.

Identificador	RF2.1
Nombre	Filtro de aplicaciones
Descripción	Realiza correctamente el filtrado de las aplicaciones de las que no se desean recibir notificaciones en el cliente web

Tabla 4: Requisito Funcional RF2.1

Toggles

A continuación se muestran las funcionalidades relativas a los toggles, que permiten modificar el estado de algunos accesos del sistema.

Identificador	RF3.1
Nombre	Activar/desactivar wifi.
Descripción	Activa o desactiva el wifi en el dispositivo móvil desde el cliente web.

Tabla 5: Requisito Funcional RF3.1

Identificador	RF3.2
Nombre	Activar/desactivar bluetooth.
Descripción	Activa o desactiva el bluetooth en el dispositivo móvil desde el cliente web.

Tabla 6: Requisito Funcional RF3.2

Identificador	RF3.3
Nombre	Activar alarma del reloj.
Descripción	Activa una alarma en el reloj desde el dispositivo del móvil a la hora introducida desde la web. En caso de ser una hora anterior actual, se activara para el día siguiente, mientras que si la hora es posterior a la hora actual, será para el mismo día.

Tabla 7: Requisito Funcional RF3.3

Identificador	RF3.4
Nombre	Enviar vibración
Descripción	Enviar una vibración al dispositivo móvil con una duración de tres segundos.

Tabla 8: Requisito Funcional RF3.4

Vídeo

En este apartado se tratan las funcionalidades relativas al envío del video desde el dispositivo móvil al navegador web del cliente.

Identificador	RF4.1
Nombre	Solicitar video.
Descripción	Solicita la demanda del video desde la cámara frontal del dispositivo móvil en caso de poseerla, en caso contrario enviará el video de la cámara posterior. La aplicación web deberá poder mostrar el vídeo en tiempo real.

Tabla 9: Requisito Funcional RF4.1

Identificador	RF4.2
Nombre	Detener reproducción de video.
Descripción	Se pausara la reproducción y envío del video desde la cámara.

Tabla 10: Requisito Funcional RF4.2

Identificador	RF4.3
Nombre	Reanudar reproducción del video.
Descripción	Se reanudara el video que anteriormente ha sido pausado.

Tabla 11: Requisito Funcional RF4.3

Localización

Este apartado contiene las funcionalidades relacionadas con la localización del dispositivo.

Identificador	RF5.1
Nombre	Obtener localización.
Descripción	La aplicación web obtendrá la localización del dispositivo y la mostrará en un mapa.

Tabla 12: Requisito Funcional RF5.1

Notificaciones

Este apartado trata de la funcionalidad relacionada con las notificaciones.

Identificador	RF6.1
Nombre	Enviar notificación del móvil al servidor.
Descripción	El dispositivo deberá enviar al servidor las notificaciones recibidas en el teléfono en tiempo real.

Tabla 13: Requisito Funcional RF6.1

Identificador	RF6.2
Nombre	Enviar notificación a un dispositivo distinto individualmente.
Descripción	Enviar una notificación aislada desde la web a un dispositivo asociado al usuario que tiene que estar previamente conectado.

Tabla 14: Requisito Funcional RF6.2

Identificador	RF6.3
Nombre	Habilitar el reenvío de todas las notificaciones a otro dispositivo.
Descripción	Reenvío de las notificaciones recibidas en el servidor a otro dispositivo.

Tabla 15: Requisito Funcional RF6.3

Identificador	RF6.4
Nombre	Descartar notificaciones desde la web.
Descripción	La aplicación web permitirá al usuario eliminar notificaciones en todos los dispositivos en los que esté.

Tabla 16: Requisito Funcional RF6.4

Identificador	RF6.5
Nombre	Activar/desactivar servicio notificaciones.
Descripción	Activa o desactiva el envío de las notificaciones desde el dispositivo al servidor, permitiendo en cualquier caso el envío de las notificaciones del servidor al dispositivo.

Tabla 17: Requisito Funcional RF6.5

Identificador	RF6.6
Nombre	Crear recordatorio desde la web.
Descripción	Creará una notificación con el título y el texto introducido por el usuario desde la web.

Tabla 18: Requisito Funcional RF6.6

Base de datos

Este apartado incluye las funcionalidades relativas a las operaciones sobre la base de datos.

Identificador	RF7.1
Nombre	Almacenamiento de notificaciones no enviadas en BdD.
Descripción	Las notificaciones que no puedan ser enviadas debido a un fallo de conexión se almacenaran en una base de datos local del dispositivo a la espera de ser enviadas cuando se restablezca la conexión.

Tabla 19: Requisito Funcional RF7.1

Identificador	RF7.2
Nombre	Guardar la información de las aplicaciones del filtro
Descripción	Se almacena en la base de datos la información de las aplicaciones que componen el filtro sobre las que no se desea que se envíen notificaciones al servidor.

Tabla 20: Requisito Funcional RF7.2

Requisitos no funcionales

Esta sección trata los requisitos no relativos a la funcionalidad de la aplicación, aquellos relacionados con las restricciones en el diseño e implementación.

Identificador	RNF1
Nombre	Precisión de la localización.
Descripción	Se obtendrá la localización más precisa posible, siendo la de mayor precisión la obtenida a través del GPS y la de menor la obtenida a través de la triangulación de las señales móviles.

Tabla 21: Requisito No Funcional RNF1

Identificador	RNF2
Nombre	Recursos necesarios.
Descripción	La aplicación móvil deberá instalarse y ejecutarse en un móvil con sistema operativo Android versión 4.3 (JellyBean) o superior. La aplicación web funcionara en cualquier cliente web con soporte para JavaScript.

Tabla 22: Requisito No Funcional RNF2

Identificador	RNF3
Nombre	Seguridad.
Descripción	Para poder usar la aplicación y la web el usuario debe estar debidamente registrado y conectado.

Tabla 23: Requisito No Funcional RNF3

Identificador	RNF4
Nombre	Tiempos de respuesta reducidos.
Descripción	Los tiempos de espera de los servicios deberían ser lo más reducidos posibles. Para la localización, inferior a 30 segundos. El envío de notificaciones, inferior a 10 segundos.

Tabla 24: Requisito No Funcional RNF4

Identificador	RNF5
Nombre	Mantenibilidad.
Descripción	El mantenimiento de la aplicación y la web deben ser sencillos de realizar. Para ello debe comentarse adecuadamente el código, modularizarlo, usar patrones de diseño que faciliten la implementación y la realización de la documentación necesaria, manual de usuario, manual de la aplicación, etc.

Tabla 25: Requisito No Funcional RNF5

Apariencia

Subsistema que trata de los elementos relativos al diseño de los clientes móvil y web.

Identificador	RNF6.1
Nombre	Facilidad de uso.
Descripción	La aplicación y la web deben ser de fácil utilización para el usuario, sin necesidad de tener que realizar muchas investigaciones para conseguir lograr su objetivo.

Tabla 26: Requisito No Funcional RNF6.1

Identificador	RNF6.2
Nombre	Interfaz agradable.
Descripción	La interfaz debe ser agradable para el usuario, con colores que faciliten la lectura de los textos, botones visibles a primera vista, tamaños de letra legibles sin esfuerzo.

Tabla 27: Requisito No Funcional RNF6.2

Casos de uso

En este apartado se realiza una descripción de los diagramas de los casos de uso del sistema, en los cuales se especifican las acciones disponibles para el usuario. En los diagramas se detallan tanto los pasos que hay que hacer para realizar una tarea como una breve descripción de lo que implica realizarla.

Dispositivo móvil

En este subsistema se muestra el caso de uso relacionado con la aplicación móvil.

Filtrado de aplicaciones

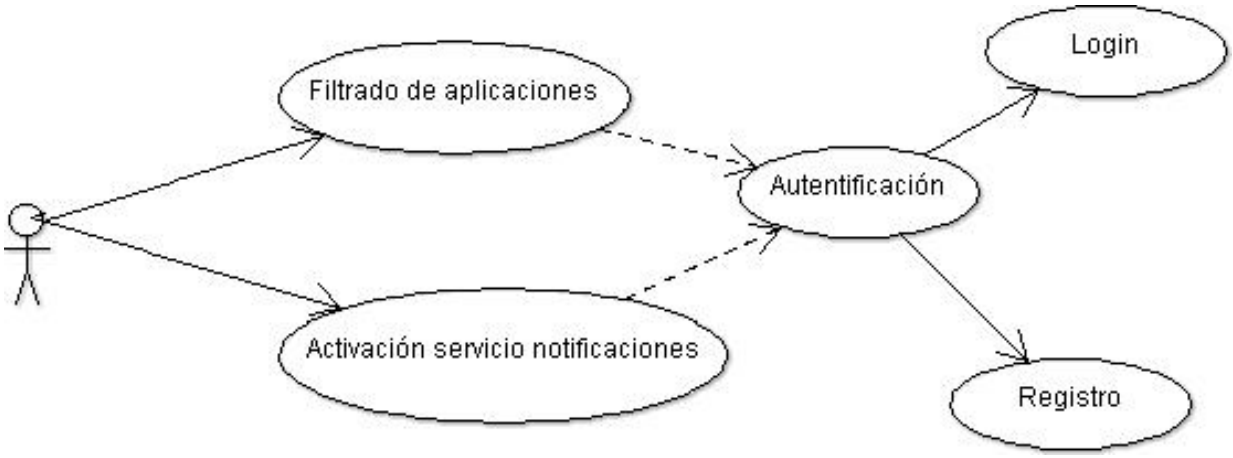

Figura 1: Caso de uso aplicación móvil

Este caso de uso, como se muestra en la figura 1, contiene las dos acciones principales de las que dispone el usuario al utilizar la aplicación móvil. Para acceder a ellas es necesario que se encuentre autenticado previamente, por lo que deberá registrarse o conectarse en caso de disponer de un usuario.

- **Filtrado de aplicaciones**: acción que permite especificar las aplicaciones que tendrán permiso para enviar las notificaciones que generen al servidor. Para indicarlas, se muestra un listado con todas las aplicaciones del sistema y al pulsar sobre cada una de ellas ésta se marca/desmarca para indicar su disponibilidad posterior.

- **Activación del servicio de notificaciones**: las notificaciones generadas por las aplicaciones instaladas en el sistema se capturan mediante un servicio que se despierta cuando se produce algún evento de notificación, tanto cuando llegan como cuando se destruyen. Este servicio puede ser activado o desactivado por el usuario.

Cliente web

A continuación se muestran los casos de uso relacionados con las acciones que se realizan en el cliente web.

Notificaciones

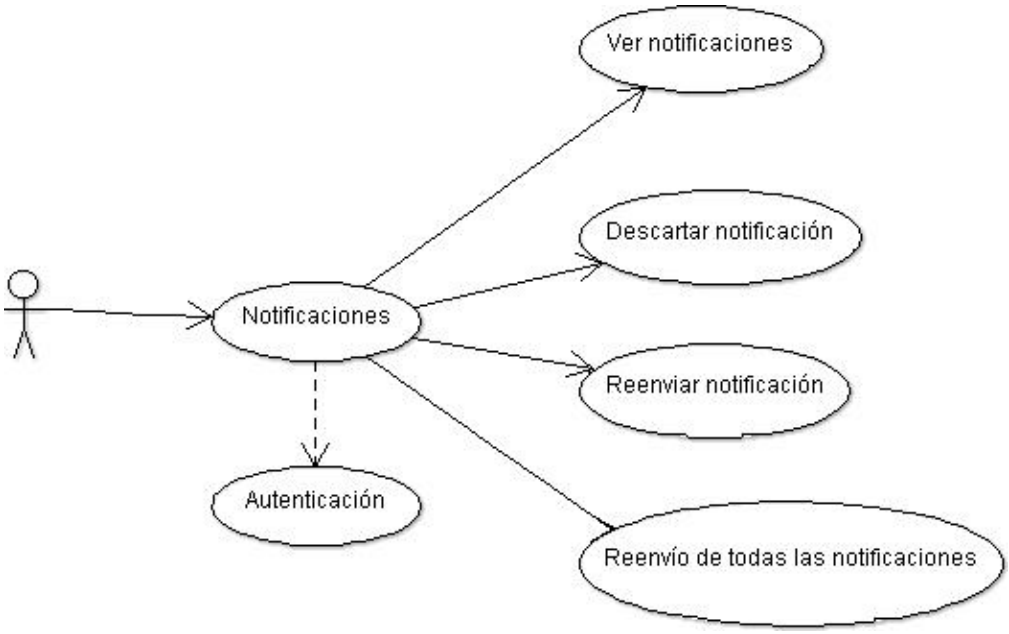

Figura 2: Caso de uso notificaciones cliente web

El caso expuesto en la figura 2 contiene las tareas que se pueden realizar para gestionar las notificaciones que van siendo recibidas en el servidor. Para ello es necesario que el usuario este autenticado previamente.

- Ver notificaciones: el usuario accede a la vista donde se muestran las notificaciones. Se muestran todas las notificaciones que recibe el servidor separadas por los distintos dispositivos que posee el usuario.

- Descartar notificación: acción que permite seleccionar una notificación y eliminarla de la lista de las disponibles en la vista. Se elimina tanto del lado del cliente web como del dispositivo móvil.

- Reenviar notificación: acción que permite reenviar la notificación seleccionada a uno de los dispositivos de los que dispone el usuario.

- Reenvío de todas las notificaciones: tarea que permite reenviar todas las notificaciones que se produzcan en un dispositivo a todos los dispositivos que se marquen en la configuración.

Servicios disponibles en la aplicación Android

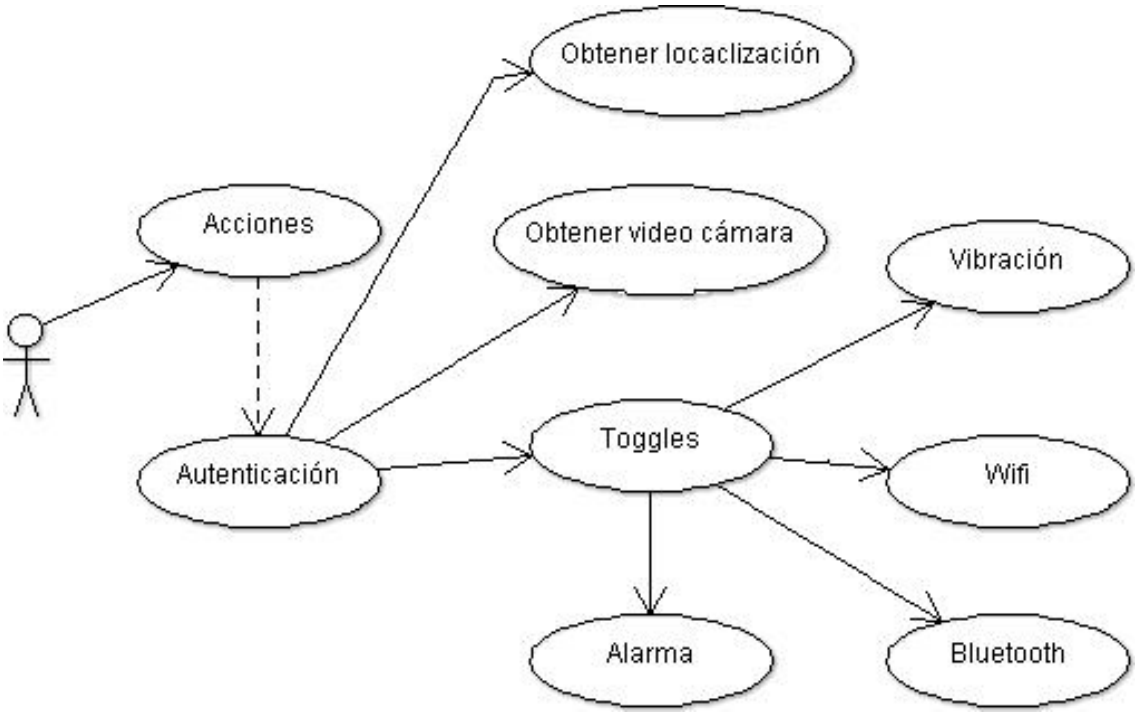

Figura 3: Caso de uso acciones cliente web

La figura 3 muestra las distintas posibilidades con las que puede interactuar el usuario a través del cliente web para realizar solicitudes o enviar acciones al dispositivo correspondiente. Para tener acceso a estas propiedades el usuario debe estar previamente conectado.

- Obtener localización: es la acción de solicitar la posición en la que se encuentra en ese instante el dispositivo.

- Obtener vídeo cámara: el usuario activa la reproducción del video y recibe directamente en el navegador, sin pasar por el servidor, el video capturado por el dispositivo móvil en tiempo real.

- Alarma: permite crear una nueva alarma indicando la hora a la que debe sonar.

- Vibración: el usuario envía una vibración al dispositivo móvil de 3 segundos de duración.

- Wifi: Acción que permite activar o desactivar el wifi.

- Bluetooth: acción que permite activar o desactivar el bluetooth.

Bloque III - Tecnologías empleadas

El proyecto hace uso de numerosas tecnologías que permiten facilitar o mejorar su realización. En esta sección se habla de ellas comentando alguna de sus características.

Tecnologías móviles empleadas

Android [1]

Los sistemas operativos más utilizados a nivel mundial son Android e IOS, y en menor escala Windows Phone y Blackberry.

En España, los datos de venta reflejados por el Kantar Worldpanel ComTech para el periodo comprendido entre noviembre de 2014 y enero de 2015, sitúan los *smartphones* con sistema operativo Android en un 86,7% de las ventas, mientras que IOS alcanza un 10,4%.

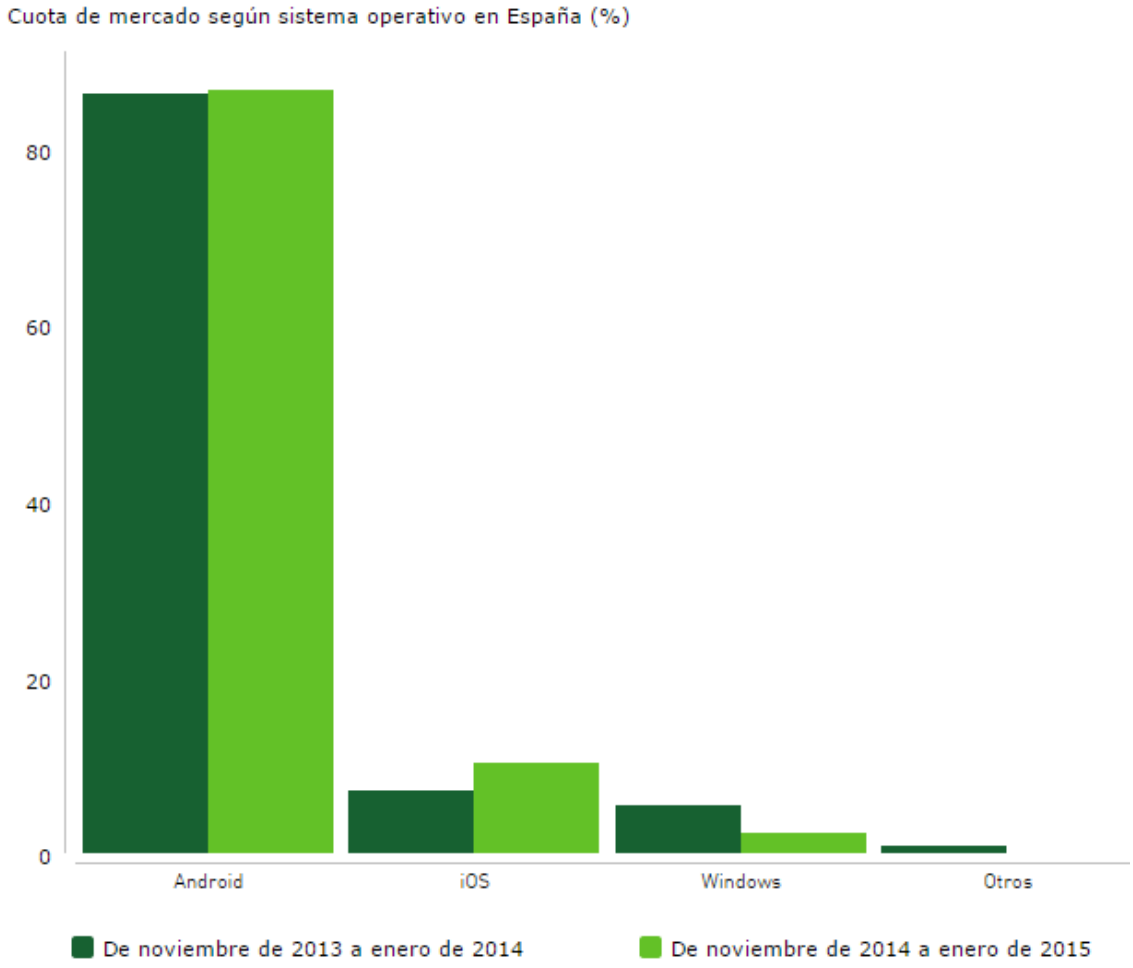

Figura 4 Gráfica cuota mercado Android en España

En Europa, los porcentajes de venta siguen mostrando un dominio de Android frente a IOS y otros sistemas operativos, aunque no tan marcado como en España, con un 67,2% de ventas.

[1] https://www.android.com/

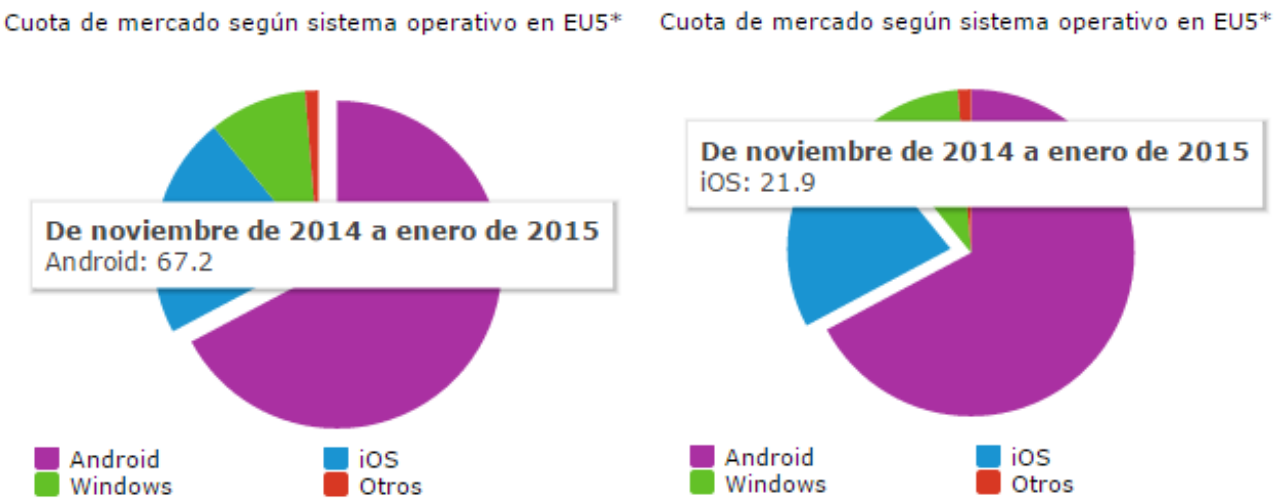

Figura 5: Cuota mercado Android EU5 2015* [2]

Como se puede observar en las figuras 4 y 5, Android es el sistema operativo más vendido en España y en Europa, y es el motivo por el que el proyecto se ha centrado en él.

Para la instalación de la aplicación desarrollada en el proyecto se requiere la versión de Android 4.3 o posterior. El motivo es debido a que se introdujo en el API la posibilidad de enlazar un servicio al sistema para escuchar las notificaciones. Esta versión se encuentra disponible actualmente en más del 50% de los dispositivos que hacen uso de Android tal y como se refleja en la figura 6 con datos obtenidos el 1 de Junio de 2015.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

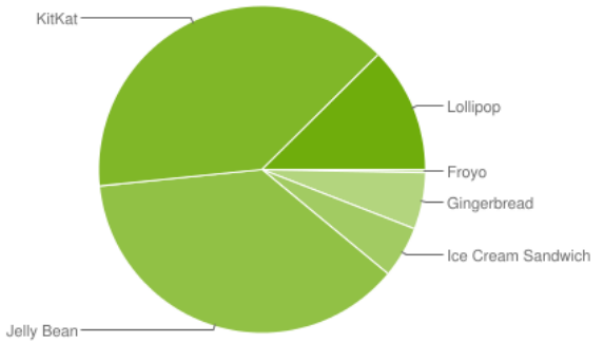

Figura 6: Uso de Android 4.3+[3]

[2] http://goo.gl/BAP5TF
[3] https://developer.android.com/about/dashboards/index.html

Crosswalk[4]

Crosswalk Project es un proyecto que permite implementar características de navegadores web en la aplicación móvil que se está creando. La gran ventaja que ofrece este sistema es que no es necesario que el usuario tenga instalado ningún navegador web en su dispositivo, ya que todos los componentes necesarios serán introducidos en la aplicación.

Los requisitos que imponen la incorporación de este proyecto es el uso de la versión de Android 4.X o superior. Como ya se había establecido que la versión compatible con la aplicación será a partir de 4.3 no supone ninguna restricción y no limita su utilización.

Gracias a este sistema, la aplicación desarrollada en este proyecto permite hacer uso del API webRTC para retransmitir videos captados desde la cámara del dispositivo móvil, a tiempo real, al navegador web del cliente.

SQLite[5]

SQLite es un sistema de gestión de bases de datos contenida en una biblioteca de ligero tamaño que se incluye en todas las distribuciones de Android. Se usará para almacenar la información local de la aplicación.

Tecnologías web empleadas

PHP 5.3[6]: Hypertext Pre – Processor

El PHP es un lenguaje muy común para programación de código del lado del servidor diseñado especialmente para el desarrollo de sitios web dinámicos, que puede ser incrustado en el código HTM. La diferencia con otros lenguajes de programación consiste en que su ejecución se realiza en el servidor, no en el navegador.

PHP permite establecer conexiones con numerosas bases de datos, incluyendo MySQL, de la que hablaremos más adelante, y se puede ejecutar en la mayoría de servidores web (incluso Apache y IIS), al funcionar tanto como módulo como procesador de CGI.

En el diagrama que a continuación se presenta en la Figura 7, se resumen las ventajas de la programación PHP, entre las que nos encontramos:

- Es un lenguaje libre y abierto, lo que permite reducir costes.
- Tiene una curva de aprendizaje muy baja, es flexible y de fácil acceso a las bases de datos permitiendo un acceso múltiple.
- Los entornos de desarrollo son de rápida y fácil configuración.
- Posee una gran comunidad, lo que garantiza un buen soporte.

[4] https://crosswalk-project.org/
[5] https://www.sqlite.org/
[6] http://php.net/

Figura 7 Ventajas uso PHP[7]

CSS 3[8]

El CSS u Hojas de Estilo en Cascada consistente en un código de programación para la presentación de estilos de páginas web de formatos HTML, XML y XHTML. Se basan en definir bases de diseño, reglas, patrones, estilos, etc. que serán la imagen del contenido de la página web que se mostrará en teléfonos móviles, tablets, portátiles, y en cualquier otro dispositivo capaz de mostrar dichas páginas. Para ello, la estructura de contenidos de la página se separa de su presentación.

JavaScript

El lenguaje de programación JavaScript, que interactúa con el código HTML, se utiliza para desarrollar sitios web más interactivos y dinámicos, ya que permite a sus desarrolladores crear acciones en sus páginas. Es un lenguaje que funciona del lado del cliente, lo que implica que no requiere compilación, que permite crear efectos e interactuaciones sin necesidad de comunicarse con el servidor del proyecto web.

Los navegadores web modernos interpretan su lenguaje, que es de código libre, sin necesidad de instalar otros programas, y en cualquier sistema operativo, ya sea Windows, Linux, Mac OS, etc.

En este proyecto se han empleado dos librerías JavaScript:

- AJAX (Asynchronous JavaScript and XML): Sistema empleado para realizar peticiones al servidor y esperar su respuesta sin necesidad de recargar la página completa.

- jQuery: Framework JavaScript que posee una gran integración con AJAX y que dota de gran dinamismo a las interfaces web.

WebRTC[9]

Es una tecnología creada por W3C (World Wide Web Consortium) que ofrece a aplicaciones móviles y navegadores web realizar comunicaciones en tiempo real para compartir datos de video y audio entre navegadores o realizar teleconferencias de par a par sin necesidad de instalar plugins.

[7] http://webdsdemo.com/wds-php
[8] http://www.w3.org/Style/CSS/
[9] http://www.webrtc.org/

Es un proyecto de software libre mantenido por Google, Mozilla y Opera con la intención de convertirse en un nuevo estándar que amplíe las capacidades del navegador web.

Esta tecnología es la base de la retransmisión de videos de la aplicación móvil del proyecto, como ya se explicó anteriormente.

WebRTC ofrece en su API tres funciones principales:

- getUserMedia: componente que permite a un navegador web acceder a los dispositivos de entrada mutimedia.

- PeerConnection, es el componente encargado de realizar la conexión entre los navegadores web. WebRTC no necesita de servidores intermedios para transmitir el contenido, pero para conectar los pares sí que es necesario el uso de un servidor. El mecanismo empleado por PeerConnection se basa en el uso del protocolo ICE[10], con el que se determina la IP pública y el puerto del usuario y después esos datos son devueltos al navegador. En algunas ocasiones el firewall oculta la dirección pública, en esos casos hay que emplear un servidor como TURN[11] o STUN[12] que permiten que puedan ser recibidos los datos en el cliente. En este caso se ha usado un servidor de Google que ponen a disposición de los usuarios.

- DataChannels, que permiten a los navegadores compartir datos a través de peer-to-peer.

Para este proyecto se ha usado como base el ejemplo desarrollado por Muaz Khan: https://github.com/muaz-khan/WebRTC-Experiment/tree/master/webrtc-broadcasting

GCM[13]

El GCM o Google Cloud Messaging, es un Sistema de notificaciones push cuya finalidad es el envío de datos de servidores a aplicaciones Android. El sistema utilizado son las notificaciones push. Este servicio controla todo lo relacionado con el almacenamiento en cola de los mensajes y su entrega a las aplicaciones.

Gracias al control de este servicio del almacenamiento en cola y la entrega de datos a los dispositivos permite que ésta última se pueda realizar a aplicaciones móviles que puedan estar en cualquier estado: activo por el usuario, en un segundo plano, en reposo o si aún no se ha arrancado.

MySQL[14]

MySQL es un sistema de almacenamiento de bases de datos relacional con servidor multihilo de software libre, al que pueden acceder aplicaciones desarrolladas con diferentes de lenguajes de programación. Este sistema permite usarse como un servidor al que se pueden conectar y consultar al mismo tiempo multitud de usuarios a la vez. Es un sistema muy empleado entre los usuarios por su simplicidad y notable rendimiento.

[10] http://en.wikipedia.org/wiki/Interactive_Connectivity_Establishment
[11] http://wikitel.info/wiki/TURN
[12] http://wikitel.info/wiki/STUN
[13] https://developers.google.com/cloud-messaging/
[14] https://www.mysql.com/

Bloque IV - Arquitectura del sistema

Debido a que tanto Android como el cliente Web se basan en una interfaz visual que interactúa con el usuario, el diseño seguido es el Modelo Vista Controlador, adaptado a las particularidades de esta aplicación. Se usa este patrón gracias a que permite separar la lógica de negocio de la interfaz de usuario, incrementando la flexibilidad y reusabilidad, permitiendo abstraer los distintos módulos entre sí.

Según el paradigma del MVC, la función de realizar gestiones sobre la base de datos dentro de la aplicación es llevada a cabo por el modelo. La asociación entre la capa de presentación y la capa de la lógica de negocio representa la integración entre la vista y el controlador encargado de los eventos y del acceso a los datos. La forma en la que este patrón intenta separar la vista del controlador permite una mejora en el desarrollo y mantenimiento tanto de la vista como del controlador.

El **modelo** es el encargado del almacenamiento de los datos, en este caso, en la base de datos de la que se hablará más adelante. Al estar separada de la vista permite la modificación de los tipos de datos sin que esta última varíe para el usuario final.

La **vista** es la encargada de mostrar la interfaz al usuario. Los datos que pueda necesitar, para obtener la máxima abstracción, se los proporciona el controlador formateados según sea necesario. Se encarga también de notificarle los eventos iniciados por el usuario al controlador.

El **controlador** es el encargado de obtener y modificar los datos del modelo, transformarlos y enviárselos a la interfaz, así como tratar los eventos recibidos desde la interfaz.

En la figura 8 se muestra el diseño seguido en el proyecto, separándolo en cuatro secciones: parte dedicada al servidor, cliente web, gestor *GCM* y la aplicación Android. Como se puede observar tanto el servidor como la aplicación siguen el diseño MVC.

La función del servidor es resolver todas las peticiones que realizan tanto el usuario a través del navegador web, como la aplicación móvil. La vista en este caso es el propio cliente web que muestra el contenido procesado por el controlador del servidor. La tecnología empleada para su desarrollo es PHP para la parte del controlador y MySQL para el modelo. La gestión del video se realiza empleando WebRTC.

El cliente hace uso de CSS, HTML y JavaScript para su desarrollo. Las dos primeras son empleadas para mostrar el contenido y JavaScript para realizar las peticiones y actualizar el contenido cuando sea necesario haciendo uso de las librerías de AJAX y jQuery.

Para que el servidor pueda enviar mensajes o responder a las peticiones que le pueda hacer la aplicación móvil se emplea un servicio intermedio, Google Cloud Message, que recibe los mensajes y los reenvía al dispositivo seleccionado.

La aplicación móvil tiene como objetivo realizar las acciones recibidas a través de GCM y enviar las notificaciones. Esta desarrollada en Android. Utiliza Java para la parte encargada del controlador y SQLite para el modelo. Las tecnologías empleadas para la gestión del video son Crosswalk para poder tener un navegador embebido y poder emplear WebRTC.

Un ejemplo de cómo es el flujo, al usar el proyecto, es el siguiente: Un usuario quiere recibir la ubicación de su dispositivo. Para ello accede al cliente web y selecciona la opción de recibir la ubicación. Ésta es enviada al controlador del servidor empleando una petición jQuery, que la procesa y consulta en el modelo (MySQL) el identificador del dispositivo al que debe enviar la solicitud de la acción. El controlador envía el mensaje con la acción y el identificador al servicio GCM que lo reenvía al dispositivo móvil. La

aplicación Android recibe el mensaje, lo procesa, realiza la acción de obtener la ubicación y la envía al servidor. El controlador del servidor recibe la respuesta, actualiza el modelo con los datos de la nueva ubicación y se la envía al cliente web que realizó la petición inicial. El cliente web procesa la ubicación y en caso de ser válida la envía al API de Google Maps y refresca la parte correspondiente de la web, usando AJAX.

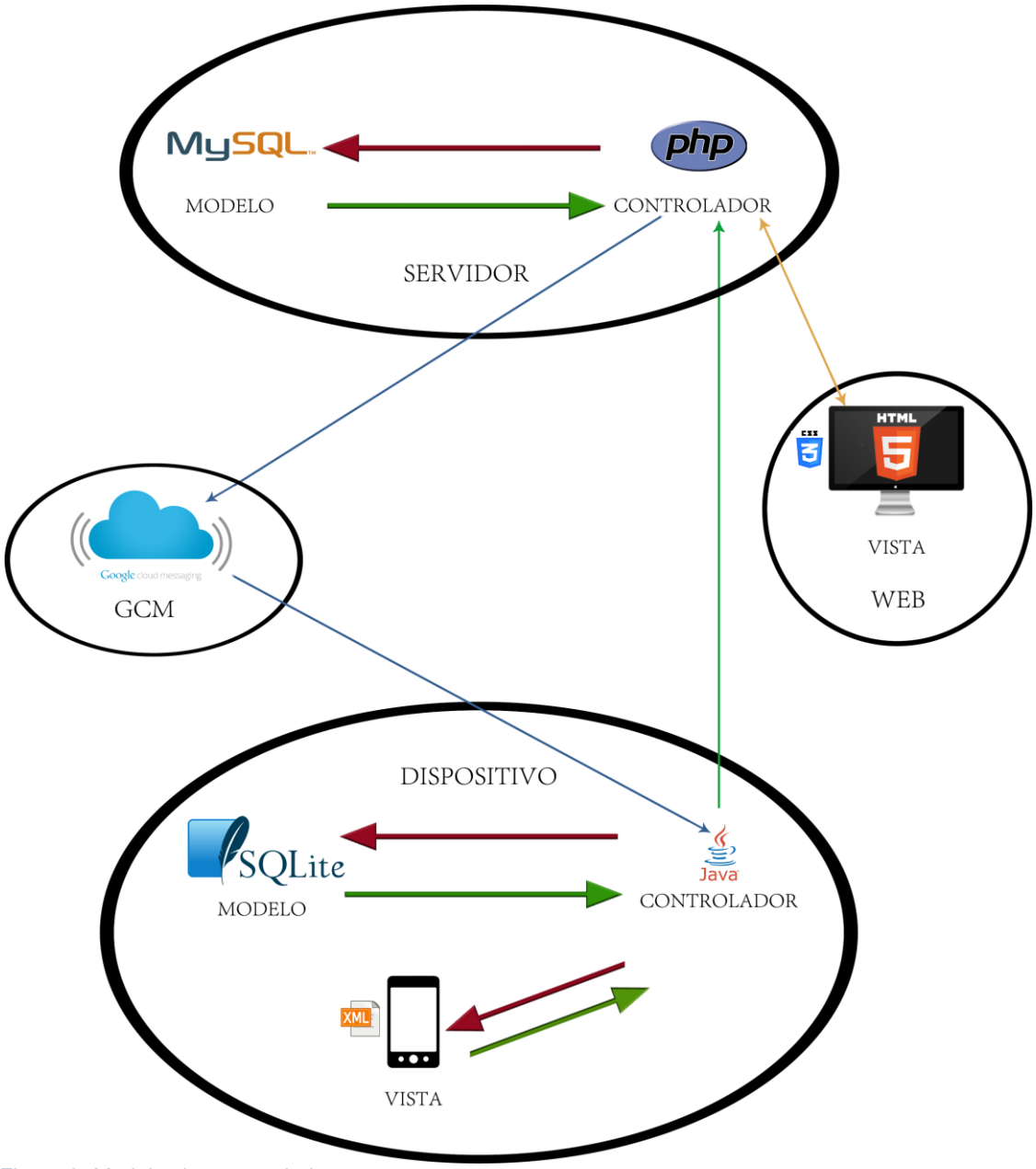

Figura 8: Modelo vista controlador

Bloque V - Fase de diseño y desarrollo

Esta sección muestra detalles sobre el desarrollo de la aplicación Android y del servidor web.

Diseño de la aplicación móvil

En la Figura 9 se muestra la estructura de la aplicación Android, incluyendo el esquema organizativo de los paquetes y sus clases.

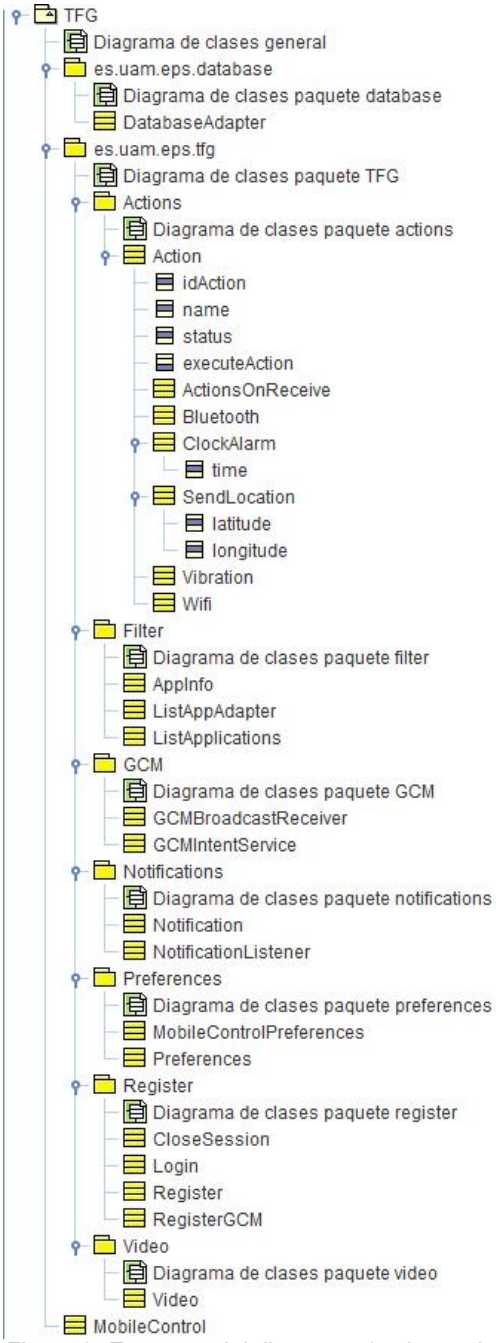

Figura 9: Esquema del diagrama de clases de la aplicación móvil

En relación a los paquetes, su organización consta de dos paquetes principales. El primero, denominado **database,** es donde se realizan las operaciones sobre la manipulación de los datos, y corresponde al modelo del patrón MVC seguido.

El segundo paquete, nombrado **tfg**, realiza las acciones propias del controlador y la vista, puesto que en Android es más complicada su separación total. Éste está formado a su vez por los siguientes subpaquetes:

- **Actions:** Incluye las clases encargadas de realizar las operaciones relativas a los eventos recibidos desde la web por parte del usuario.

- **Filter:** Realiza el filtrado de las aplicaciones del dispositivo que no deberán enviar notificaciones contra el servidor.

- **GCM:** Módulo encargado del servicio Google Cloud Messaging que realiza la escucha de los mensajes recibidos y los transmite a la clase encargada de tratarlos.

- **Notifications:** Este es el paquete en el que se encuentra el servicio anclado al sistema capaz de procesar todas las notificaciones recibidas en el dispositivo. Cuando se genera o recibe una notificación en el sistema se despierta este servicio, la recibe y la procesa.

- **Preferences:** Almacena datos de preferencias relativos al usuario.

- **Register:** Se encarga de los servicios de registro y login del usuario en el servidor.

- **Video:** Envía el video capturado por la cámara del dispositivo al navegador del usuario.

En la figura 10 se muestra el diagrama de clases de la aplicación incluyendo los paquetes expuestos. Veamos cada módulo con más detalle.

Database

- **DatabaseAdapter:** clase encargada del control con la base de datos, realizando las escrituras y lecturas sobre ella.

tfg.actions

- **ActionsOnReceive:** Esta clase es la encargada de distribuir las acciones. Recibe un mensaje con el evento de la acción y lo envía al método que corresponda.

- **Action:** Clase abstracta con la definición de la funcionalidad básica de las acciones a realizar. En ella se almacenan el identificador de la acción, su nombre y la acción a realizar sin implementar. Las clases comentadas a continuación son subclases que heredan de esta.

- **Bluetooth:** Clase encargada de activar y desactivar el bluetooth.

- **ClockAlarm:** Clase que activa una alarma del reloj a la hora especificada en el mensaje.

- **SendLocation:** Obtiene la localización del dispositivo, comprobando primero el método disponible para obtener la posición con mayor precisión, siendo el chip de GPS el de mayor exactitud y los datos móviles con posición a través de triangulación de señales los más imprecisos. Una vez conseguida se envía al servidor.
- **Vibration**: activa una vibración de tres segundos de duración.

- **Wifi:** se encarga de activar y desactivar el wifi.

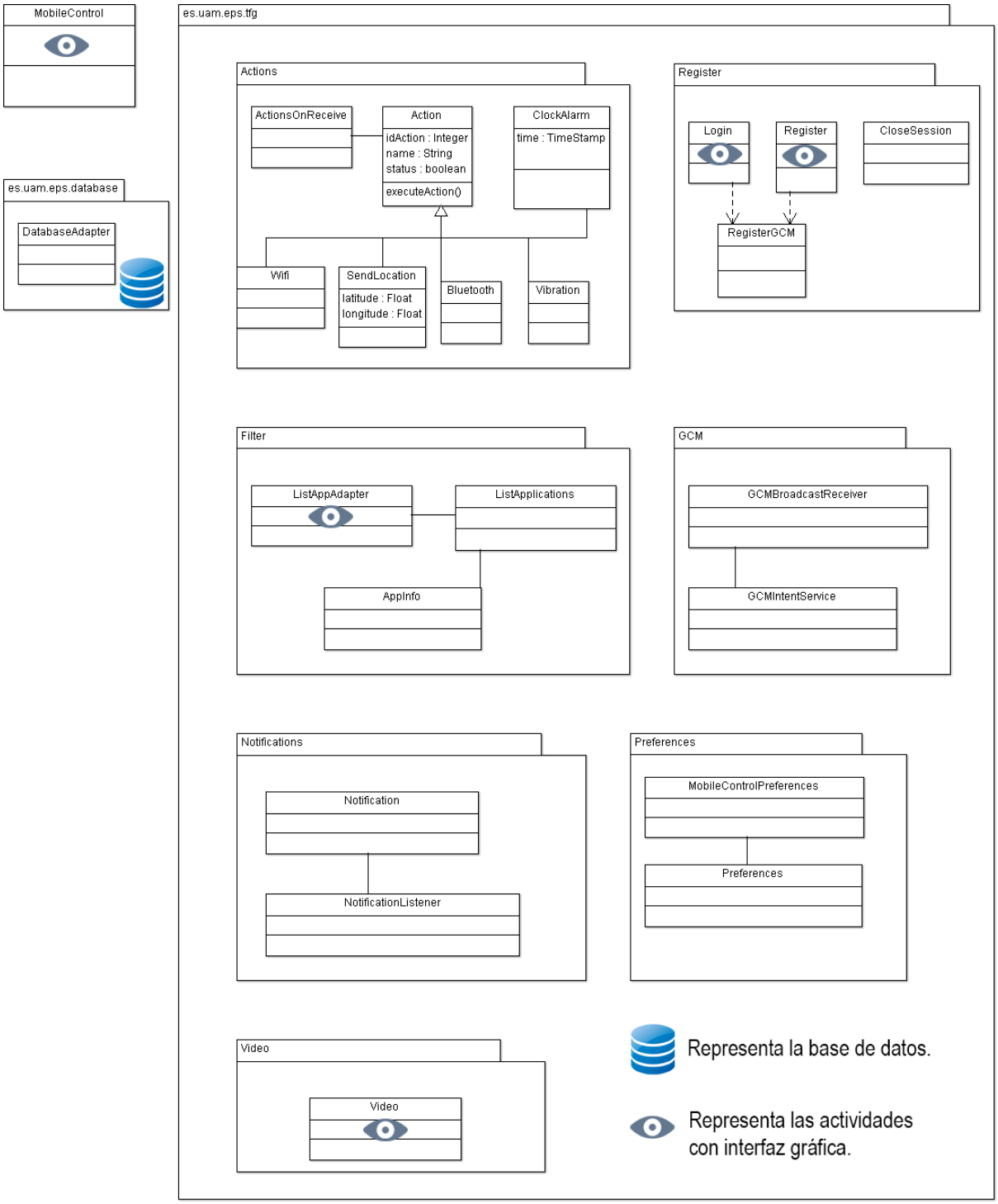

Figura 10: Diagrama de clases de la aplicación

tfg.filter

- **AppInfo:** contiene la información necesaria para identificar unívocamente una aplicación del sistema, almacenando su identificador, título, imagen y descripción.
- **ListApplications:** es el encargado de generar la lista de las aplicaciones para crear un filtro que contenga los elementos que no deben enviar notificaciones al servidor. Para ello obtiene todas las aplicaciones que hay disponibles en el sistema, usando AppInfo como contenedor de información de cada una de ellas.

  Una vez obtenida la lista completa comprueba en la base de datos (DatabaseAdapter) cuales son las aplicaciones que se marcaron previamente para ser filtradas.

  Para poder añadir una aplicación al filtro se captura cada elemento de la lista de aplicaciones y en caso de ser presionado este es añadido o eliminado de la base de datos.
- **ListAppAdapter:** crea el adaptador gráfico con las aplicaciones generadas por ListApplicactions.

tfg.gcm

- **GCMBroadcastReceiver:** clase de tipo BroadcastReceiver que recibe los mensajes push enviados a la aplicación desde el servidor. Mantiene el dispositivo activo durante el tiempo que dure la acción asociada al mensaje recibido.
- **GCMIntentService:** servicio llamado por GCMBroadcastReceiver para procesar el mensaje recibido. Comprueba su integridad y lo descompone para notificárselo a ActionsOnReceive. En el momento en el que la acción se ha realizado realiza el dispatch del servicio generado por GCMBroadcastReceiver.

En la figura 11 se muestra el flujo por el que pasa una petición enviada a través de *GCM* desde que se crea la acción hasta que termina su ejecución.

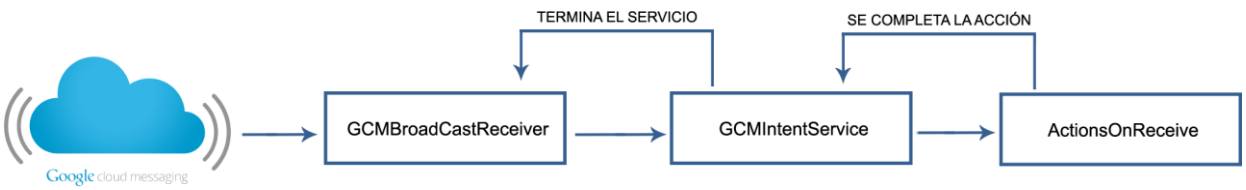

Figura 11: Flujo acción

tfg.notifications

- **Notification:** almacena la información básica con los elementos de una notificación: identificador y etiqueta únicos, contenido del mensaje a mostrar, nombre de la aplicación que la generó y su paquete.

- **NotificationListener:** servicio ejecutado cuando una notificación es recogida por el sistema. La notificación es procesada y comprobada por el filtro de aplicaciones. Si es admitida se envía al servidor.

  En caso de que se produzca un error en el envío, como que no esté disponible el acceso a la red en ese instante, la notificación es almacenada en la base de datos para ser reenviada cuando sea posible.

tfg.preferences

- **MobileControlPreferences:** actividad que permite editar preferencias del usuario.

- **Preferences:** fragmento que almacena información asociada al usuario. Incluye información de acceso frecuente como los datos de registro o el identificador de *GCM* para comprobar su validez.

tfg.register

- **CloseSesion:** se encarga de finalizar la sesión cuando es requerido por el usuario. Borra todos los datos almacenados en la base de datos y envía una petición al servidor para que finalice la sesión web, de tal forma que si accedemos a la web con ese usuario, los datos de ese dispositivo no sean mostrados.

- **Login:** realiza la conexión del usuario en la aplicación realizando una petición al servidor. Llama a RegisterGCM en caso de que el login se realice con éxito.

- **Register:** registra al usuario en el servidor y realiza el login del usuario en caso de que el registro se complete con éxito.

- **RegisterGCM:** realiza el registro del dispositivo en el servicio de mensajería de Google.

tfg.video

- **Video:** envía el video de la cámara al navegador del usuario que realiza la petición a través del cliente web. Para ello hace uso del proyecto de *Crosswalk Project* mencionado anteriormente.

Diseño del servidor

Este apartado muestra una descripción del modelo de datos del servidor y de las funciones que componen el cliente web, así como de algunos detalles de su implementación.

Diseño de la base de datos

Esta sección contiene información relativa al método de almacenamiento de datos. El cliente web hace uso de una base de datos relacional MySQL compuesta de las tablas que se detallan a continuación. La figura 12 muestra el diagrama entidad-relación.

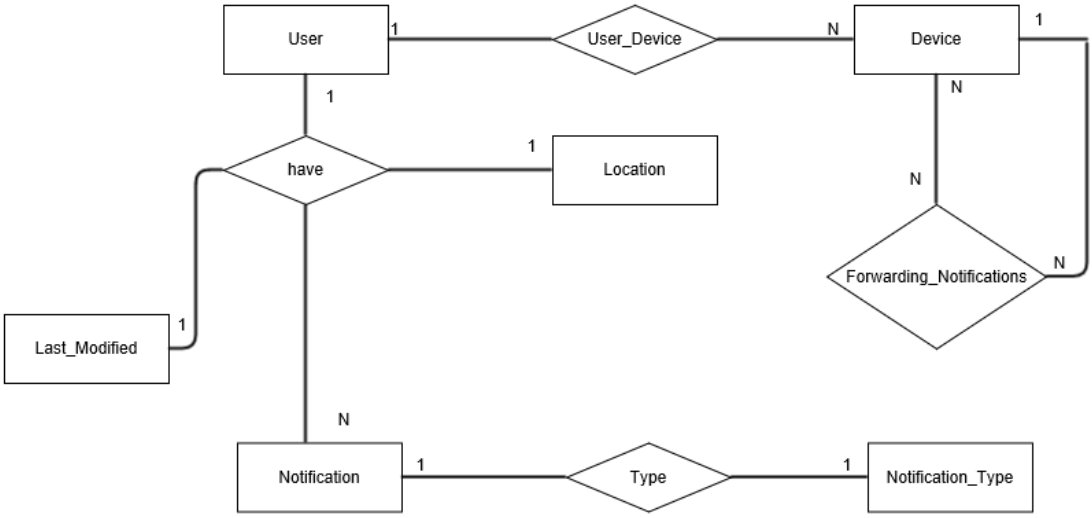

Figura 12: Diagrama E/R

Users

Nombre de la tabla	USERS	
Atributos y tipo	Nombre atributo	Tipo atributo
	- user_id	- serial
	- user_name	- char (50)
	- user_pass	- char (100)

Tabla 28: Users

La tabla users almacena los usuarios que se registran empleando la aplicación. Almacena el nombre del usuario, “user_id” y su contraseña, “user_pass” codificada en md5. El identificador “user_id” se genera incrementalmente por cada nuevo usuario para poder mantener la integridad en todas las tablas con las que se relacione el usuario aunque haya modificaciones posteriores.

Clave primaria: user_id, identificador del usuario.

Devices

Nombre de la tabla	DEVICES	
	Nombre atributo	Tipo atributo
	- device_id	- serial
	- device_gcm	- text
	- user_id	- character integer
	- device_name	- character (50)

Tabla 29: Devices

Almacena los dispositivos registrados en el sistema para cada usuario (user_id). Contiene un identificador autoincremental del dispositivo "device_id" que se asocia con el nombre del dispositivo, "device_name", que es el nombre del modelo que tiene asociado con el sistema operativo y el identificador "device_gcm", empleado por el servicio *GCM* para poder enviar un mensaje push al dispositivo. Este código es único, pero es posible que con el paso del tiempo sea necesario actualizarlo, lo que impide que sea un candidato de clave primaria.

Clave primaria: device_id, identificador del dispositivo.

User_Devices

Nombre de la tabla	USER_DEVICES	
	Nombre atributo	Tipo atributo
	- device_id	- integer
	- user_id	- integer

Tabla 30: User Devices

Relaciona los usuarios con los dispositivos. Un usuario puede tener varios dispositivos, pero el mismo dispositivo no puede estar disponible para más de un usuario,

Clave foráneas: "device_id", hace referencia a device_id de la tabla devices, y user_id hace referencia a la tabla users.

Notification_Type

Nombre de la tabla	NOTIFICATION_TYPE	
	Nombre atributo	Tipo atributo
	- noti_id	- serial
	- noti_package	- char(50)
	- noti_name	- char(50)
	- noti_img	- char(50)

Tabla 31: Notification Type

Para evitar el exceso de datos en los mensajes enviados por el dispositivo móvil al servidor, se ha creado la tabla "notification_type" empleada para almacenar las imágenes de las aplicaciones que generan notificaciones. Estas imágenes son los *launcher icons*. La tabla asocia cada aplicación identificándola mediante su nombre "noti_name" y el paquete al que pertenece "noti_package" con la ruta relativa a su icono "noti_img".

Clave primaria: noti_id, identificador de la aplicación.

Notifications

Nombre de la tabla	NOTIFICATIONS	
	Nombre atributo	Tipo atributo
	- id	- serial PRIMARY KEY
	- noti_type_id	- int
	- device_id	- text
	- user_id	- char(50)
	- noti_read	- boolean
	- noti_text	- text
	- noti_id	- text
	- noti_tag	- text
	- noti_date	- Timestamp

Tabla 32: Notifications

Almacena las notificaciones recibidas por el servidor. Una notificación se conforma por su identificador, compuesto por “noti_id” y “noti_tag” y su contenido, almacenado en “noti_text”. También se incluye información relativa al estado de la notificación, indicando si ésta ha sido leida, “noti_read”, y la fecha en la que fue entregada “noti_date”

Clave primaria: id, identificador de la notificación.

Clave foránea: noti_type_id, hace referencia a Notification_type, device_id hace referencia a la tabla devices y user id a users.

Last modified

Nombre de la tabla	LASTMODIFIED	
	Nombre atributo	Tipo atributo
	- id	- serial
	- user_id	- char(50)
	- last_modified	- Timestamp

Tabla 33: LastModified

Almacena cuándo se ha producido la última acción por parte del usuario. Esta tabla es actualizada cada vez que se recibe un evento en el servidor y es empleada para consultar cuándo se ha registrado una nueva notificación.

Clave foránea: user_id, hace referencia a la tabla users.

Forwanding Notifications

Nombre de la tabla	FORWARDING_NOTIFICATIONS	
	Nombre atributo	Tipo atributo
	- id	- serial
	- sender_id	- bigint(20)
	- receiver_id	- bigint(20)

Tabla 34: Forwarding Notifications

Crea las relaciones que indican a qué dispositivos reenviar las notificaciones. “Sender_id” es el identificador del dispositivo desde el que reenviar las notificaciones y “receiver_id” el destinatario. Un dispositivo puede reenviar las notificaciones a varios.

Clave foránea: sender_id y receiver_id hacen referencia a devices.

Locations

Nombre de la tabla	LOCATIONS	
	Nombre atributo	Tipo atributo
	- id	- serial
	- latitude	- text
	- longitude	- text
	- last_update	- Timestamp
	- device_id	- Bigint(20)
	- user_id	- Char(50)

Tabla 35: Locations

Almacena la última localización recibida del dispositivo. La localización es expresada mediante la longitud, "longitude", y latitud "latitude", y se relaciona con el dispositivo mediante su identificador. El campo de la última actualización, "last_update", es empleado para conocer en qué momento se ha recibido la última localización

Clave foránea: device_id hace referencia a la tabla devices y user_id a users.

API del servidor

Login

Página encargada tanto de crear la sesión del usuario como de cerrarla. Para ello hace uso de los datos almacenados en la tabla *users* de la base de datos. Esos datos deben haber sido introducidos previamente en la aplicación registrando un nuevo usuario mediante la página de registro.

Registro

Las páginas encargadas de realizar el registro de los dispositivos y de los usuarios son las siguientes:

- **DeviceRegister**: añade un dispositivo al sistema. Para ello asocia el id generado por *GCM* que permite recibir las notificaciones en el dispositivo con el usuario. Este método no es solo llamado en los casos en los que se da de alta un usuario, ya que el identificador generado por el servicio de GCM tiene un tiempo de expiración. Por ello también se encarga de actualizar el id por el nuevo correspondiente cuando corresponde. En lo referente a datos de usuario se encarga de inicializar los campos que representan la geolocalización del dispositivo y lo hace estableciendo la latitud y longitud a cero.

- **Register**: se encarga del registro del usuario almacenando su información en la base de datos y de la inicialización de los campos que representan la última modificación del dispositivo. Estos son campos que se actualizan cada vez que se realiza una acción relacionada con las notificaciones.

- **LoginMobile**: realiza las comprobaciones pertinentes para responder al dispositivo si los datos del usuario para realizar la conexión son correctos.

Actualización de estado

Son los módulos relativos a la actualización de la base de datos en base a acciones realizadas desde el dispositivo.

- InsertLocation: actualiza los campos de longitud y latitud del dispositivo.

- ActionsNotificacions: recibe la acción a realizar sobre una notificación y los elementos que la identifican (id, paquete y tag). Las acciones pueden ser insertar o eliminar la notificación.

  En caso de que la petición sea de inserción significa que el dispositivo móvil ha enviado una notificación al servidor. La función inserta el elemento en la base de datos y comprueba la tabla de reenvíos que tiene asignada ese usuario. En caso de que en la tabla se muestre el dispositivo que envió el mensaje, éste será reenviado a todos los identificadores devueltos por la consulta.

  La acción de eliminar busca la notificación en la base de datos y la descarta.

GcmMessage

Módulo que compone el mensaje para enviar al servicio de *GCM.* El mensaje es un elemento JSON que contiene los datos del mensaje.

Desarrollo de las páginas de la aplicación web

Index

Página inicial de la web. Muestra todo el contenido relacionado con la gestión de acciones del dispositivo separado en tres secciones.

La figura 13 muestra cómo la página inicial donde se realizan los cambios de estado.

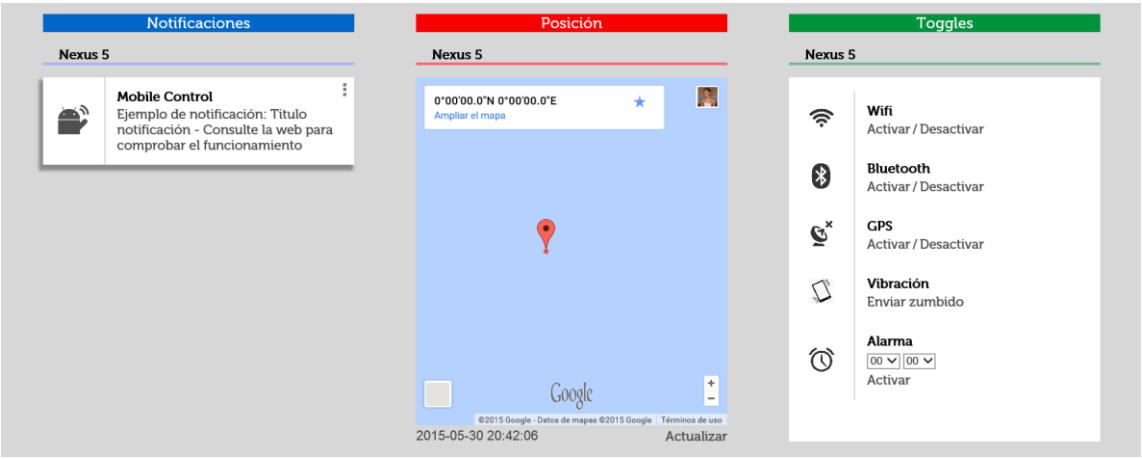

Figura 13: Página de inicio

La primera sección es la encargada de mostrar todas las notificaciones recibidas en el servidor que son enviadas desde los dispositivos. Esta sección se refresca de forma asíncrona mediante AJAX cada vez que se produce una nueva notificación en la base de datos. La figura 14 muestra un ejemplo de cómo se muestran las notificaciones cuando son recibidas.

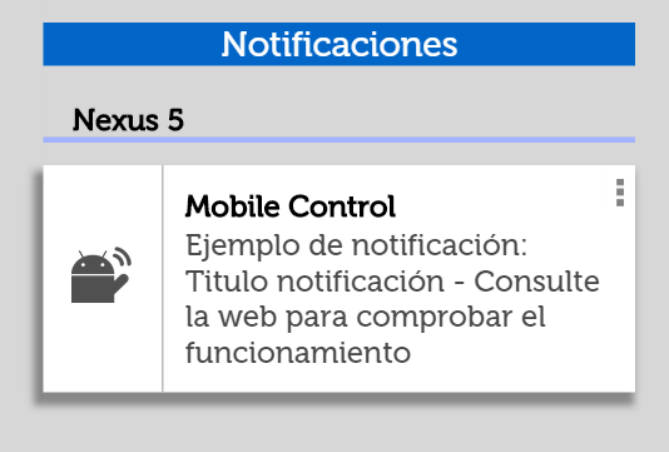

Figura 14: Ejemplo de notificación

Para detectar cuando hay una nueva notificación el servidor se emplea la técnica de *long polling [2]* (ver Figura 15).

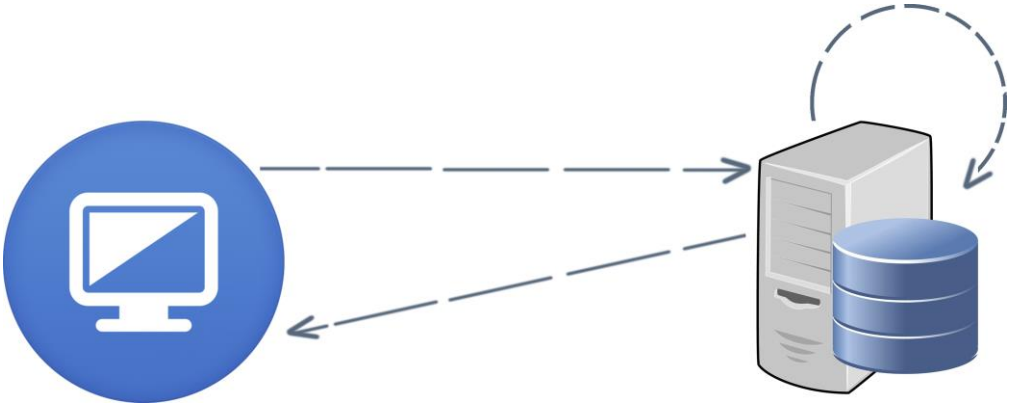

Figura 15: Long Polling

Esta técnica es una variación del método de polling. En este caso en lugar de realizar una petición y recibir una respuesta directamente, el cliente mediante una petición post de jQuery realiza una consulta al servidor para comprobar si hay una notificación nueva disponible. El servidor procesa la petición y en caso de que la respuesta sea negativa, en lugar de responderle, se guarda la consulta y espera a que haya disponible nuevos elementos. La conexión entre el cliente y el servidor se mantiene abierta hasta que termine el proceso.

El servidor realiza una espera entre las consultas para impedir el exceso de procesamiento. Para impedir que el proceso se quede huérfano, debido a que el cliente haya decidido cerrar la sesión, hay un timeout que en caso de alcanzarse cierra la conexión. Si salta el timeout y el cliente sigue estando operativo, recibirá la respuesta de que no hay ninguna notificación nueva y volverá a mandar la petición al servidor.

Esta forma de consultar las notificaciones permite que no haya peticiones constantes que sobrecarguen la red y la recepción por parte del servidor.

La segunda sección es la relacionada con la localización de los dispositivos. En este caso la solicitud se produce bajo demanda del usuario de la aplicación web. Cuando realiza la petición para obtener la ubicación, ésta es enviada al dispositivo que la procesará y enviará una respuesta de nuevo al servidor. Para comprobar si el dispositivo ha respondido se emplea también el método de *Long Polling*, pero en este caso si salta el timeout no se vuelve a realizar la petición y se le indica al usuario que no ha sido

posible obtener la localización. La figura 16 muestra cómo se representan las ubicaciones recibidas en la web.

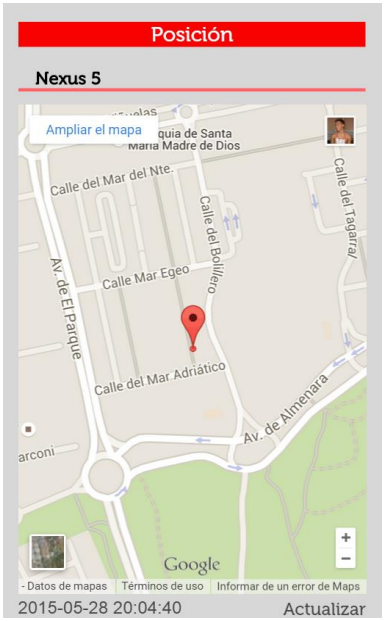

Figura 16: Ejemplo de ubicación

Los datos recibidos representan la longitud y latitud y son mostrados haciendo uso del servicio de *Google Maps [3]* en caso de que la respuesta sea válida.

La tercera sección muestra las funcionalidades relacionadas con el envío de peticiones al dispositivo móvil para activar o desactivar *toggles* del sistema. Para ello al ser pulsado uno de estos elementos envía un mensaje empleando *GcmMessage* expuesto en el apartado anterior. En el mensaje se indica el nombre del toggle sobre el que realizar la acción y atributos necesarios para su funcionamiento, como por ejemplo, la hora en el caso de la alarma. La figura 17 muestra cómo se representan las acciones disponibles en la página web.

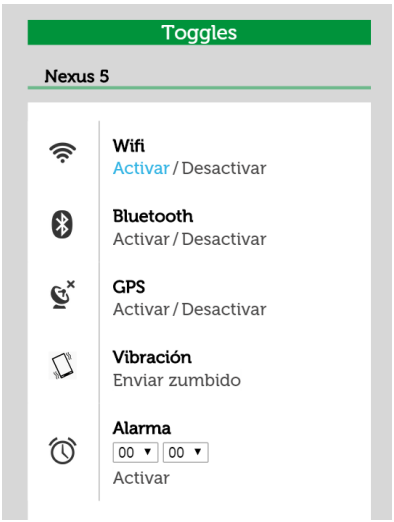

Figura 17: Ejemplo de toggles

Video

Pantalla que realiza la petición de video para recibir el contenido capturado por la cámara del dispositivo móvil en streaming. Para ello se hace uso de *WebRTC[5].* El dispositivo móvil se conecta a la página SendVideo en la que se incluyen las funciones para su envío. La figura 18 muestra la pantalla de solicitud de video.

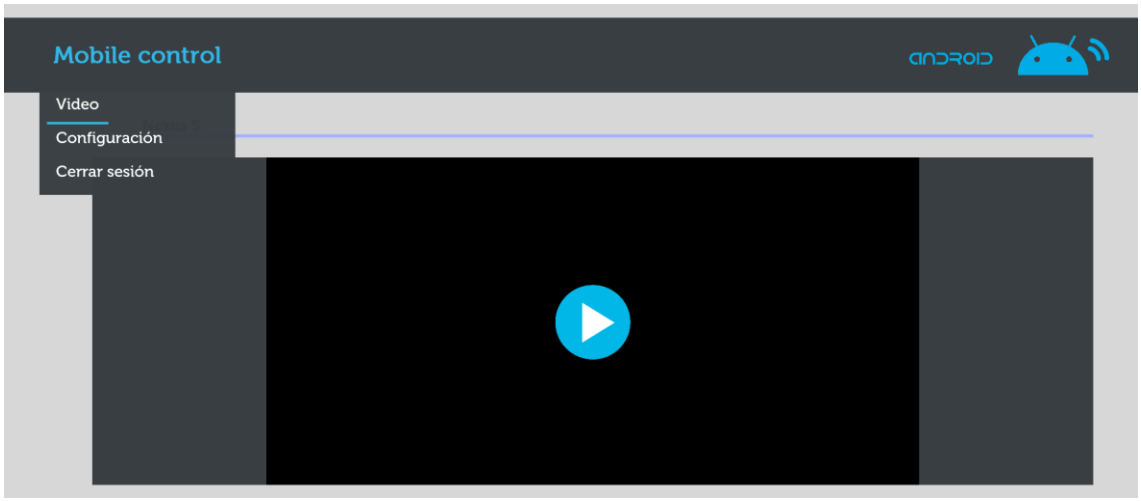

Figura 18: Ejemplo solicitud video

Bloque VI - Pruebas

Esta sección describe el plan de pruebas realizado a lo largo del desarrollo del proyecto así como los resultados obtenidos. El objetivo de esta fase es abarcar el mayor número de situaciones en las que se podría encontrar el usuario para poder detectar y mitigar el mayor número de errores posible. De esta forma se busca intentar asegurar el funcionamiento de la aplicación según los casos expuestos en el análisis de requisitos como se presenta en la figura 19.

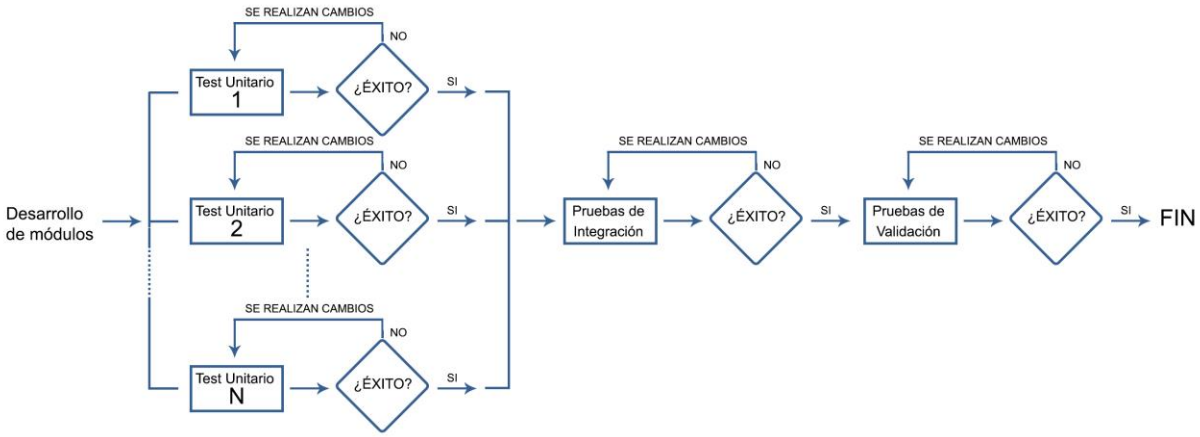

Figura 19: Diagrama de pruebas

Pruebas unitarias

Pruebas que se realizan para comprobar el correcto funcionamiento de los módulos del código por separado. Comprueba tanto la lógica como la funcionalidad especificada.

Estas pruebas han sido realizadas de forma manual durante todo el desarrollo del proyecto debido a que el esfuerzo de localizar los errores en esta fase es menor por tener menos dependencia de otros módulos.

Algunas de las pruebas realizadas:

- Control del funcionamiento de las entradas de datos por separado: tanto en la web como en la aplicación móvil se ha testeado el comportamiento de los botones y de los diálogos de entrada de texto, comprobando que se produzcan los efectos esperados observando los resultados que se van obteniendo en los listeners asociados a cada botón.

- Prueba de cada una de las acciones: el dispositivo realiza varias acciones, como activar el bluetooth, el wifi… cada uno de estos métodos han sido probados por separado.

- Prueba del módulo ActionsOnReceive: se ha simulado la entrada de datos para comprobar que la acción que determina qué se debe realizar sea la correcta y en caso de reciba una acción que no esté implementada en el sistema, la ignore y termine el servicio.

- Componer mensaje para GCM: se comprueba que el método encargado de componer los mensajes que se enviarán usando el servicio de *GCM* sigan una estructura correcta validando el JSON generado.

- Creación de la base de datos interna de la aplicación (SQLite): se comprueban los métodos dedicados a la creación y actualización de la base de datos. Para ello se analizan los scripts SQL encargados de realizar las actualizaciones dividiéndolos según la instrucción que deben realizar para, posteriormente, comprobar su ejecución.

Pruebas de integración

Pruebas que afectan al uso de varios módulos independientes que deben interactuar entre ellos. Tanto en la parte móvil como en la parte web la mayor parte de situaciones requieren de la cohesión de varios elementos. Para ello se ha probado el funcionamiento de las distintas opciones que ofrece la aplicación móvil y el cliente web por separado. Algunos ejemplos de estas pruebas:

- Ejecución de una acción una vez recibido un mensaje en la aplicación: Se prueba que el proceso de parseo del mensaje, análisis de la acción y ejecución de la misma se realicen correctamente.

- Recepción de una notificación en el lado del cliente web: se comprueba que una vez que se recibe una notificación la página lo detecte (mediante *long polling)* y refresque la parte de la página referente a las notificaciones con la nueva información.

- Realización del registro usando el dispositivo móvil: se comprueba que los datos introducidos son válidos, que no haya conflicto de existencia previa del usuario y que los datos sean almacenados en la base de datos.

- Acceso a las preferencias para activar el reenvío de notificaciones: se selecciona un dispositivo al que reenviar las notificaciones provenientes de otro y se realiza la prueba de generar una notificación. Ésta deberá aparecer en el terminal elegido y los datos deberán mostrarse a su vez tanto en la base de datos del móvil como en la del servidor.

Pruebas de validación

Este apartado muestra las pruebas realizadas para comprobar que el producto cumple con todas las funcionalidades especificadas en el análisis de requisitos.

Se presentan las pruebas en función de las distintas situaciones en las que se puede encontrar el usuario al utilizar tanto la aplicación móvil como el cliente web. La técnica empleada es el de caja negra, ya que en este punto no es posible realizarlas en función del código programado y solo se tiene en cuenta que se cumpla la funcionalidad.

Es un apartado especialmente importante en este proyecto debido a que casi todas las funcionalidades afectan tanto a la parte de la aplicación móvil como a la del cliente web, repercutiendo uno sobre el otro.

1. **Prueba de login de usuario**: Se ha probado el caso de conexión del usuario tanto en la aplicación móvil como en el cliente web (ver figura 20).

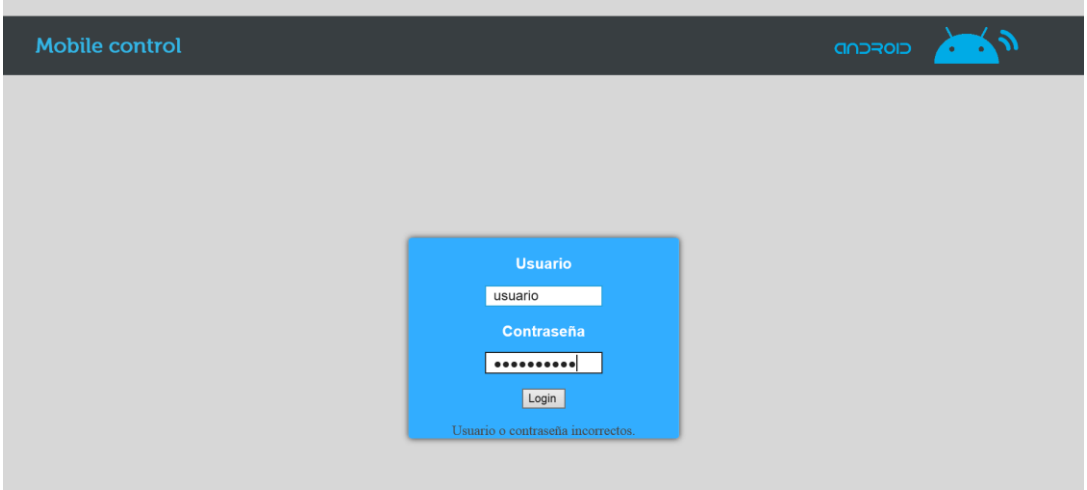

Figura 20 Login erróneo en la web

En caso de que introduzca datos de acceso inválidos es notificado y se le permite volver a introducirlos. Este apartado está relacionado con la aplicación móvil, pues si no está registrado previamente con el dispositivo no podrá realizar la conexión.

2. **Cerrar sesión:** cuando el usuario cierra la sesión en la aplicación móvil los datos asociados que tenga en el servidor deben ser eliminados, borrando los registros de la base de datos, de forma que en la página web no pueda ver el contenido (ver figura 21).

Figura 21: Cierre de sesión

3. **Crear notificación:** la aplicación dispone de un botón para probar el servicio de envío de notificaciones. Al presionarlo se genera una notificación y ésta deberá ser enviada al servidor y mostrada automáticamente en el cliente web (ver figura 22).

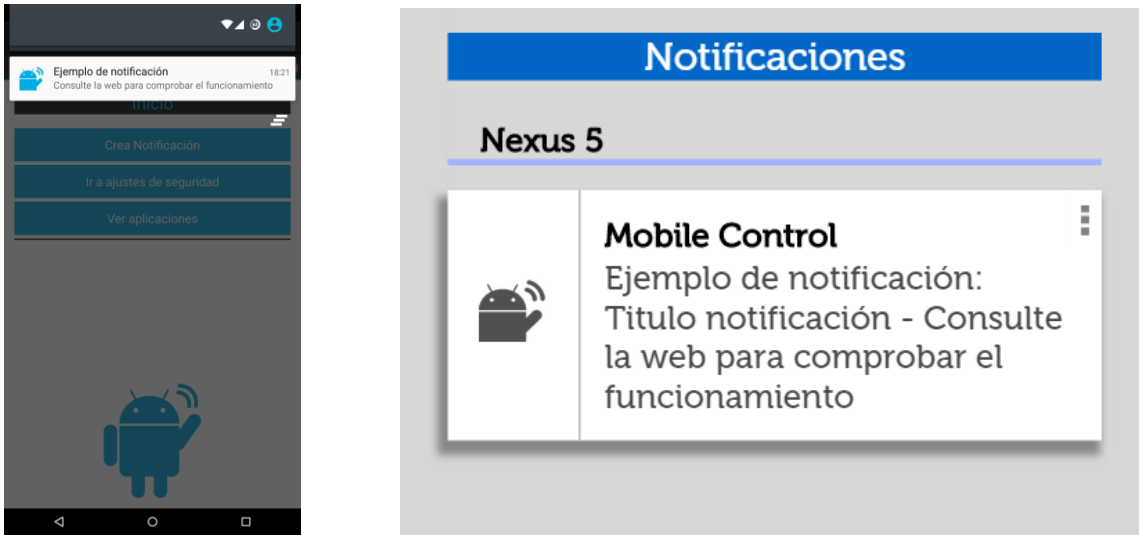

Figura 22: Crear notificación

4. **Activar un toggle del dispositivo desde la web:** en esta comprobación se procede a enviar una vibración al dispositivo (ver figura 23).

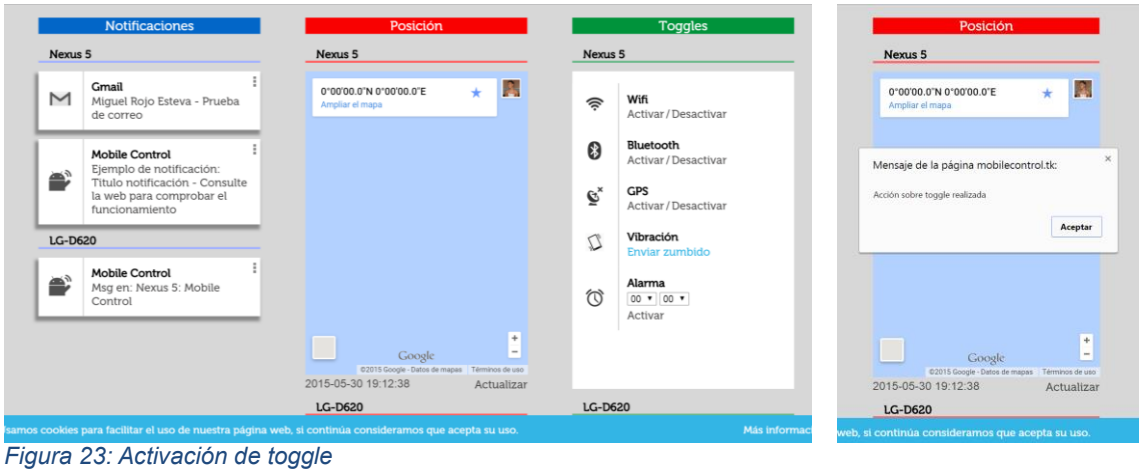

Figura 23: Activación de toggle

5. **Petición de localización:** el usuario solicita la posición del dispositivo a través del cliente web (ver figura 24).

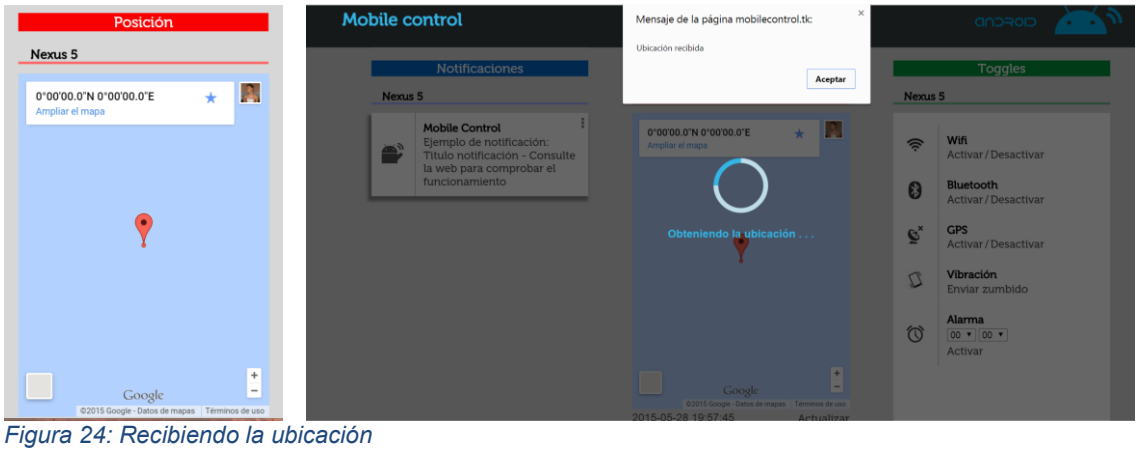

Figura 24: Recibiendo la ubicación

Se pueden dar dos situaciones:

a. El dispositivo recibe la petición y responde con su posición (ver figura 25).

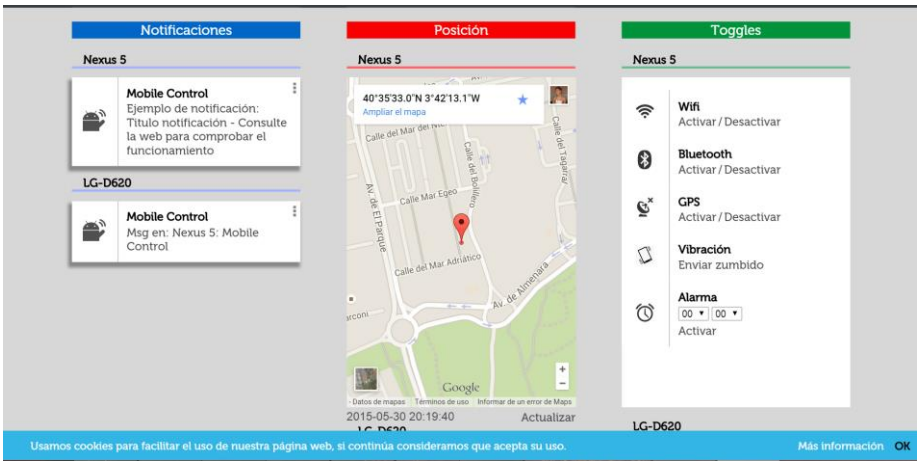

Figura 25: Ubicación recibida

b. El dispositivo no puede responder la ubicación y salta el timeout de espera para la petición (ver figura 26). Los casos que hacen que se produzca este error son debidos a que el terminal este apagado o que tenga desactivados los servicios de ubicación: no se pueden activar internamente desde la aplicación por cuestiones de privacidad.

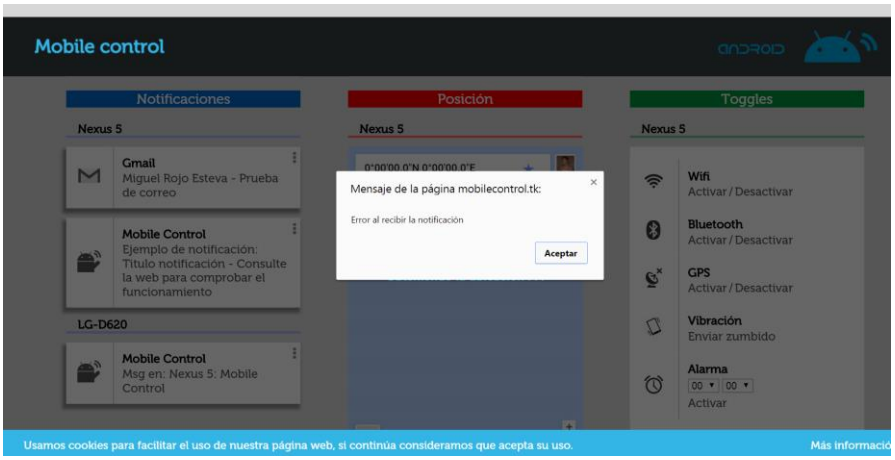

Figura 26: Error al recibir la ubicación

6. **Solicitud de video:** el usuario recibe desde la web el video capturado por la cámara del terminal en streaming (ver figuras 27 y 28).

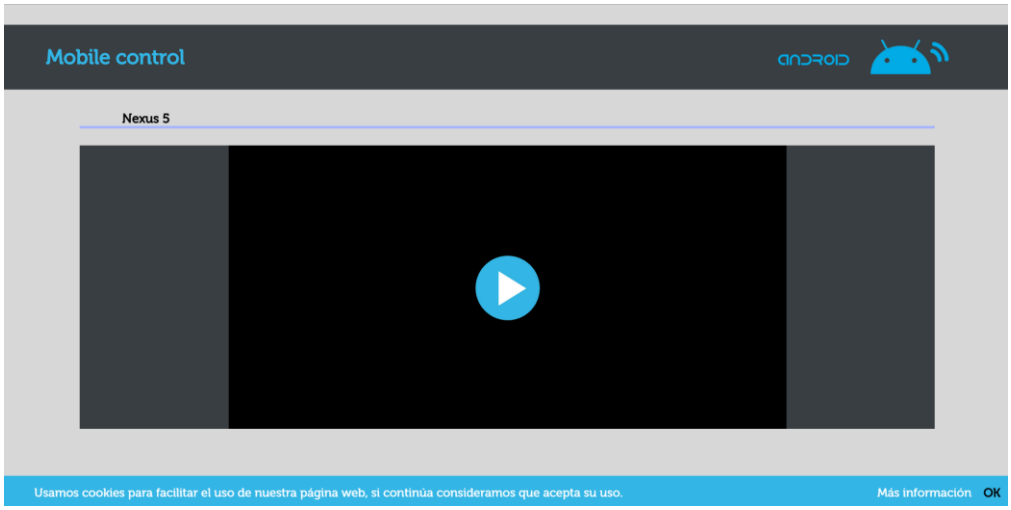

Figura 27: Solicitud de video

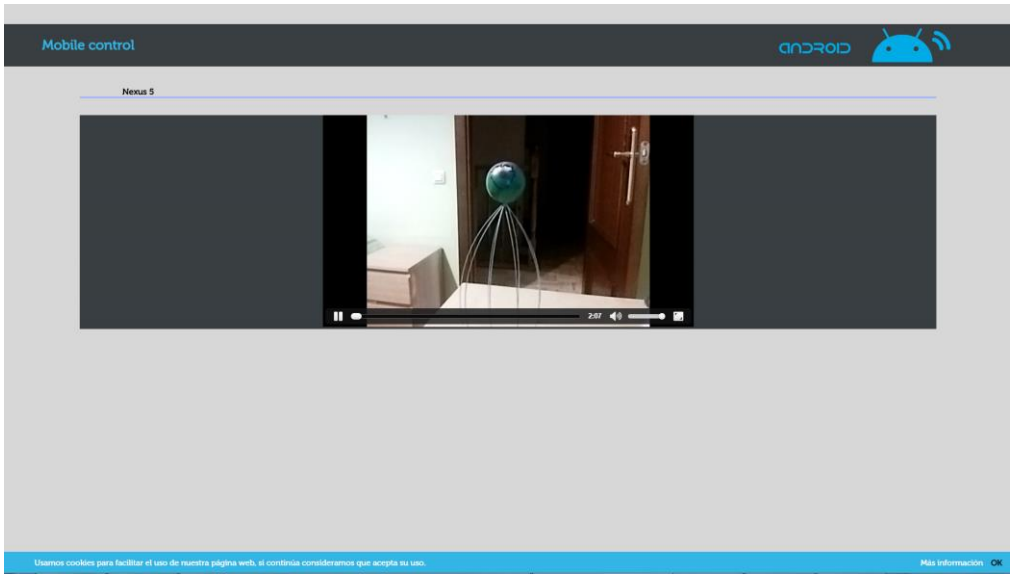

Figura 28: Reproducción de video

Bloque VII - Conclusiones y trabajo futuro

Conclusiones

El principal objetivo que se perseguía del proyecto era la realización de un sistema que permitiera obtener información relevante de varios dispositivos móviles, como son las notificaciones, de una forma unificada, clara y limpia en un cliente web y permitir a su vez la interacción con alguna de las capacidades que nos ofrecen los *smartphones* desde el cliente web.

Tras la finalización de este proyecto se ha comprobado que se han cumplido todos los objetivos expuestos en el documento y se puede hacer uso del sistema de una manera sencilla e intuitiva sin la necesidad de tener que hacer configuraciones complejas en la aplicación móvil.

En lo referente a la evolución de la realización del trabajo, la parte que considero más importante ha sido el análisis de las tecnologías a emplear. En primer lugar se realizó un análisis de los requisitos que tiene el proyecto, y de cada conjunto de estos era necesario averiguar cuál era la metodología a seguir para abordarlo. Eso ha sido una labor compleja, ya que inicialmente partía desde una visión general en la que desconocía las distintas tecnologías que se podrían emplear para resolver cada uno de los apartados. Especialmente importante ha sido el esfuerzo dedicado al uso de la tecnología WebRTC para enviar video en streaming desde los dispositivos móviles a la aplicación web de forma directa.

Gracias a este trabajo considero que podría realizar más aplicaciones relacionadas con esta temática necesitando una curva de aprendizaje bastante menor y con la capacidad de adquirir nuevos conocimientos de una forma ágil y productiva.

Trabajo futuro

En este apartado se abordan los aspectos mejorables de la aplicación, así como las funcionalidades que se podrán incluir.

Para mejorar la eficiencia en la actualización de las notificaciones en el cliente web, se propone el uso de websockets para realizar la conexión entre el servidor y el cliente, ya que de esta forma se mantiene la conexión abierta permitiendo el envío de datos bidireccionalmente. Esto implica un menor consumo de recursos y evitar tener que realizar peticiones constantemente.

En lo relativo al diseño web se actualizará para hacer uso de Bootstrap, que es un framework que permite crear interfaces usando CSS y Javascript que se adapten a los dispositivos, es decir, conformar los elementos en función del tamaño de la pantalla.

Mejorar el apartado del GPS permitiendo más modos de seguimiento: dar la opción de que el terminal envíe su localización en el momento en el que detecte que se ha desplazado o enviar a localización al cabo de un tiempo establecido.

Con esto se podría elaborar también una tabla con todas las posiciones del usuario en un intervalo de tiempo y analizar su comportamiento habitual.

El apartado del video en este instante tiene propiedades establecidas por defecto. Estas propiedades podrán ser modificadas para indicar qué cámara debe ser la que capture el video o indicar la calidad y resolución que tendrá la imagen.

Referencias

[1] http://es.wikipedia.org/wiki/MD5

[2] http://webcooker.net/ajax-polling-requests-php-jquery/

[3] https://developers.google.com/maps/tutorials/fundamentals/adding-a-google-map

[4] http://www.html5rocks.com/en/tutorials/webrtc/basics/

[5] http://www.webrtc.org/

[6] https://developer.android.com/google/gcm/index.html

[7] http://www.importancia.org/android.php

[8] https://php.net/manual/es/index.php

[9] http://www.w3schools.com/

[10] http://www.desarrolloweb.com/manuales/taller-ajax.html

[11] http://api.jquery.com/

[12] http://danielme.com/2013/12/08/diseno-android-viewpager/

[13] David Flanagan. *JavaScript: The Definitive Guide.* O'Reilly Media

[14] Jonathan Simon. *Head First Android Development.* O'Reilly Media

[15] Robin Nixon. *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media*

Apéndice I - Ciclo de vida

Este apartado se centra en el ciclo de vida tanto del proyecto como del sistema, lo que incluye a la aplicación móvil y la página web.

El ciclo de vida de un elemento es la sucesión de las distintas fases que lo componen, desde el nacimiento de la idea hasta que el producto deja de usarse en el caso de que el elemento sea un proyecto, o desde que se inicia hasta que se apaga en el caso de que sea un sistema.

Ciclo de vida del proyecto

El ciclo de vida de un proyecto parte desde el momento en el que nace la idea hasta que el producto resultante de dicha idea se deja de usar. Para llegar desde un estado a otro se atraviesan distintas fases que se irán analizando a continuación (ver Figura 29).

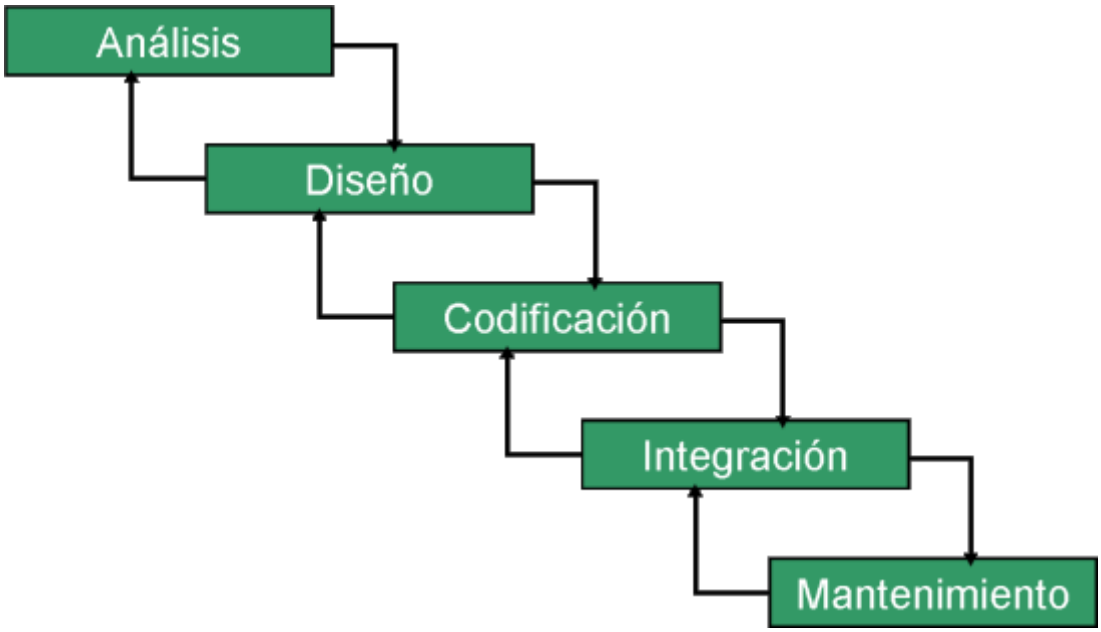

Figura 29: Ciclo de vida de un proyecto software

La fase de **análisis** es aquella en la que nace la idea del proyecto, y en la que se empiezan a sentar las bases del futuro producto. En esta fase se establecen todos los requisitos que debe cumplir el producto, tanto aquellos que recogen la funcionalidad que debe tener, como aquellos relacionados con las restricciones en el diseño e implementación. Del análisis de requisito se habla en el apartado correspondiente.

La fase de **diseño** es aquella en la que una vez capturados todos los requisitos del producto, se procede a idear el formato del producto, tanto en su parte visual, como en la parte del código. Para este proyecto en particular se diseñaron varias maquetas tanto de la aplicación móvil como de la página web. En cuanto a la parte del código, se ha modularizado separando en paquetes lo más independientes posibles cada una de las clases para la aplicación, y realizando cada funcionalidad de la web en un módulo diferente del resto.
La fase de **codificación** es aquella en la que se desarrolla el producto. Se ha realizado una programación legible, dando nombres a las variables de tal modo que sepa cuál es

su uso, y además, se ha comentado el código para facilitar su entendimiento. En esta fase se han realizado las pruebas unitarias, es decir, elemento por elemento individualmente, de las que se habla en el apartado pruebas unitarias, además de algunas pruebas de integración entre elementos individuales, para comprobar el correcto funcionamiento entre ellos, sobre lo que se habla en pruebas de integración.

La fase de **integración**, en un proyecto empresarial, sería la fase en la que se realizan las pruebas de integración sobre el sistema del cliente, pero en este caso, la fase de integración ha consistido básicamente en realizar el resto de pruebas de integración, así como ciertas pruebas de validación, que son aquellas en las que se prueba que el producto funciona en todos los sistemas finales, distintos navegadores, pantallas con resoluciones distintas, dispositivos móviles con sistemas operativos distintos, y distintas capas de personalización, etc. Sobre este tipo de pruebas se habla más detalladamente en el apartado en el apartado pruebas de validación.

La fase de **mantenimiento** aún no ha llegado a este proyecto, pero consistirá en todas aquellas mejoras que se harán sobre el sistema, así como la solución de los posibles errores que se vayan detectando.

Ciclo de vida de la aplicación Android

En este caso, el ciclo de vida de la aplicación está formado por las fases desde que se arranca hasta que es eliminada de la lista de aplicaciones en segundo plano.

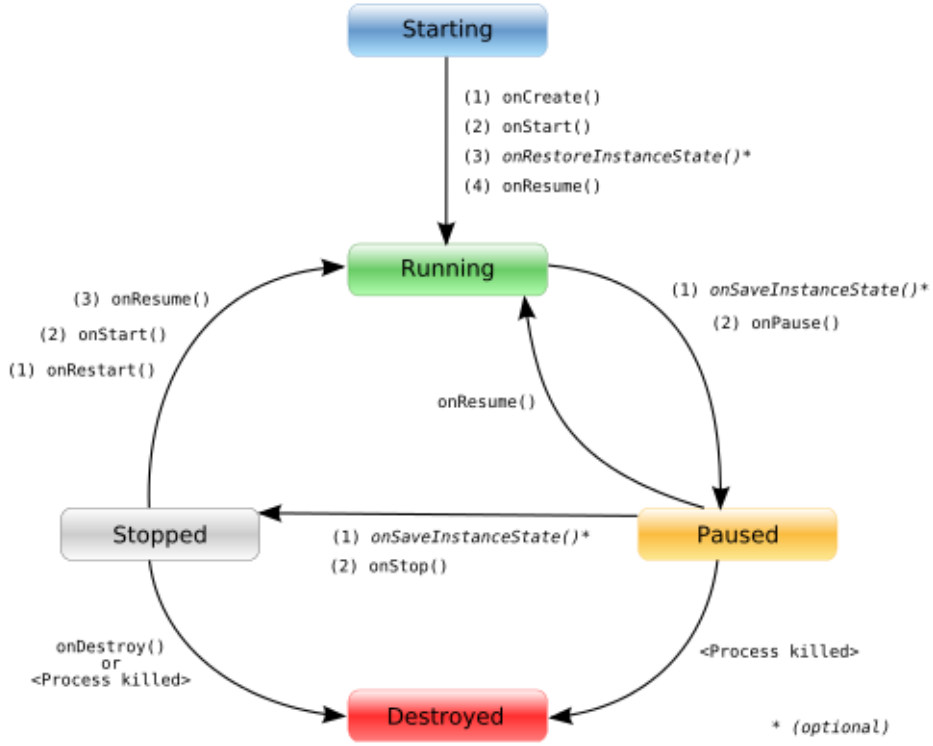

Figura 30: Ciclo de vida de una aplicación Android

Como se observa en la figura 30, una aplicación móvil consta de 5 fases principales que son:

- **Comienzo:** aquella en la que la aplicación arranca desde cero, es decir, o es la primera vez que se ejecuta, o anteriormente fue cerrada completamente.
- **En ejecución:** fase en la que la aplicación está ejecutándose en primer plano en el terminal y el usuario puede interactuar con ella.

- **Pausada:** fase en la que la aplicación sigue en primer plano, pero el usuario no puede interactuar con ella porque existe algún evento que requiere la atención del usuario.

- **Parada:** proceso en el que la aplicación pasa a estar disponible, pero en segundo plano. En este estado la aplicación puede seguir realizando tareas con el procesador y recibiendo notificaciones, pero no puede interactuar con el usuario.

- **Destruida:** aquella fase en la que la aplicación se cierra completamente y se liberan todos los recursos que tuviese asignados.

La aplicación Android nada más iniciarse por el usuario (fase de ejecución), comprueba si se ha identificado anteriormente, saltándose entonces el paso de autenticación. Sin embargo, si no se encuentra conectado, el programa muestra al usuario la pantalla en la que tiene que identificarse pudiendo simplemente autenticarse o realizar un registro para un usuario nuevo.

Una vez autenticado, el flujo continúa a la pantalla principal en la que hay tres botones, cada uno de los cuales realiza una funcionalidad distinta, como se muestra en la figura 31:

- **Botón superior:** crea una notificación de prueba en el móvil para comprobar el correcto funcionamiento. Si el sistema funciona adecuadamente, la notificación debería llegar a la página web, siempre y cuando no la borre el usuario antes de que esto ocurra.

- **Botón intermedio:** muestra la pantalla en la que el usuario debe activar la lectura de notificaciones del sistema por parte de la aplicación. Si esta opción no está activada, la aplicación no puede acceder a la lectura de las notificaciones y por tanto no podrá realizar las comprobaciones pertinentes y enviarlas al servidor.

- **Botón inferior:** muestra la pantalla del filtro de aplicaciones. En esta pantalla, se listan todas las aplicaciones disponibles en el dispositivo con su nombre y su logo, así como un icono que indica si se envían o no notificaciones sobre dicha aplicación. Por defecto están todas activadas

Figura 31: Inicio aplicación Android

Existe otra pantalla que es la de envío de video en streaming, que es la única funcionalidad que no se puede ejecutar en segundo plano.

En este punto, aunque el usuario cierre completamente la aplicación, es decir, que no esté en primer o segundo plano, la aplicación seguirá realizando la lectura y el envío de las notificaciones de las aplicaciones no filtradas, ya que el servicio está anclado al sistema y es despertado cuando se produce un nuevo evento de notificaciones.

Ciclo de vida de la página web

El ciclo de vida de la página web objeto de proyecto es similar al de cualquier página web. Desde que el usuario introduce su dirección en el navegador, hasta que cierra la página.

Nada más cargar, la página web nos muestra, al igual que ocurría en la aplicación móvil, la pantalla de identificación, en la que el usuario debe autenticarse frente al servidor.

Una vez autenticado, el flujo de la web continua hasta la pantalla principal, dividida entre tres grandes bloques:

- El bloque de la **izquierda** muestra todas las notificaciones recibidas por el servidor. Desde este bloque se pueden descartar las notificaciones, reenviar todas a un dispositivo asociado al usuario desde ese instante, reenviar una sola notificación puntual a un dispositivo asociado al usuario, así como crear recordatorios y enviarlos a los dispositivos del usuario, en los cuales aparecerá una notificación con el título y el texto que el usuario indique.

- El bloque **central** muestra la geolocalización del dispositivo mediante la API de *Google Maps*. Esta localización será lo más precisa posible según la conectividad activa en el dispositivo. Se trata de una localización en tiempo real.

- El bloque de la **derecha** contiene los toggles, como son el Wifi o el bluetooth y que pueden activarse y desactivarse a gusto del usuario. También puede configurar una alarma introduciendo la hora y minutos deseados. El funcionamiento de la alarma se discutirá más adelante. Del mismo modo, se puede activar en el dispositivo una vibración de tres segundos de duración.

Si el usuario pulsa sobre el logo de la página (arriba a la izquierda) le aparece la opción de ir a la pantalla de ver el video de la cámara en tiempo real, donde se mostrará un reproductor con la posibilidad de iniciar el servicio usando WebRTC.

Apéndice II – Google Cloud Messaging

En esta sección se exponen los pasos necesarios para poder añadir el servicio *GCM* en una aplicación Android. El manual expuesto a continuación ha sido incluido en la asignatura Desarrollo de Aplicaciones para Dispositivos Móviles para que otros estudiantes puedan añadirlo a su proyecto.

1. El primer paso es obtener la autorización de Google para el uso del servicio asociándolo a una cuenta. Para ello hay seguir las siguientes indicaciones:
   - Entrar en https://cloud.google.com/console/project
   - Pulsar en Create Project. Una vez creado hay que abrir el desplegable de la izquierda APIs & auth y después seleccionar APIs.

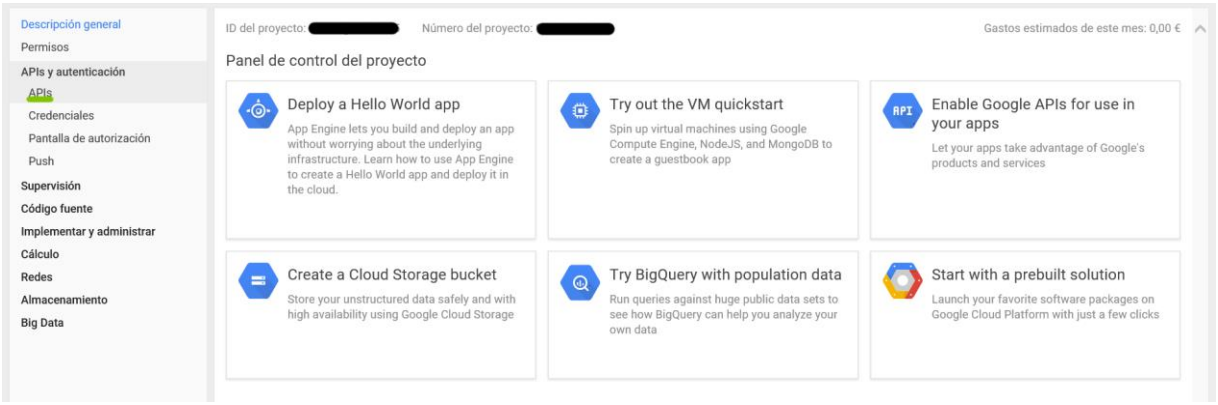

Figura 32: Apartado APIs GCM

   - Se mostrará una lista de los servicios que se pueden activar. El que interesa para GCM es el que se llama Google Cloud Messaging for Android (hay que pulsar sobre él y activarlo).
   - Una vez activado hay que obtener una clave de identificación de acceso:
     - En el desplegable de APIs & auth pulsar sobre Credentials.
     - En el apartado de Public API access -> Create new key.
     - Pulsar sobre Browser key // Create.

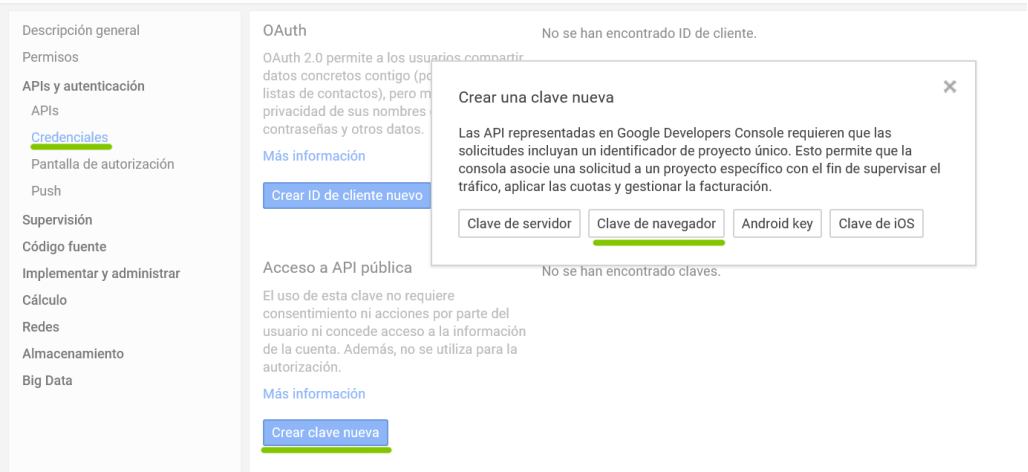

Figura 33: Generar clave navegador GCM

- Con eso se consigue el API KEY. Hay que guardar esa clave para utilizarla más adelante.

- Obtener el número del proyecto (SenderID):
  - En el desplegable de la izquierda -> Api Project
  - Seleccionar Overview
  - En la parte superior hay un campo que dice Project Number:...
  - Guardar ese número (para el campo SENDERIDGCM del paso 3)

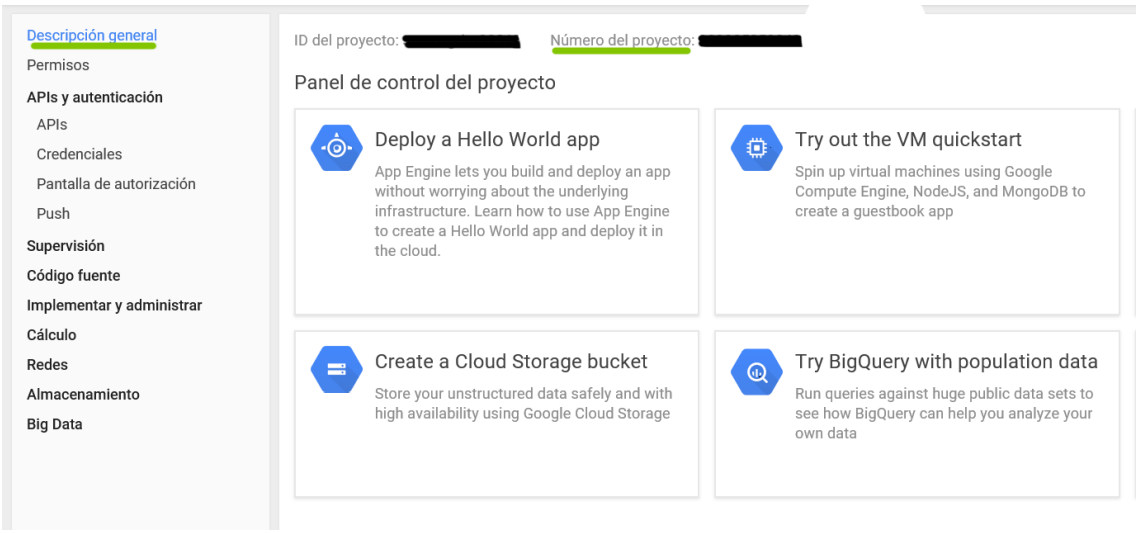

Figura 34: Número proyecto GCM

2. El segundo crear un método en el servidor para poder enviar mensajes GCM. En este caso se realiza en PHP.

   Con la siguiente función se puede enviar un mensaje a un dispositivo android. Los dispositivos android deben estar registrados en GCM (paso número 3). El mensaje $msg será enviado a todos los dispositivos que se indiquen en $listMobiles. $listMobiles es un array con todos los identificadores (cada móvil tiene su propia key -> paso 3).

```
/**
 * Envía un mensaje a la lista de móviles especificada
 * @param type msg mensaje que se desea enviar
 *  @param type $listMobiles lista de móviles a los que enviar el
 *  mensaje
 */
function enviaMensajePush($msg, $listMobiles){
      //Browser key: clave para la comunicación con el servidor de google
      message
      $apiKey = ********** ; //La clave obtenida en el paso anterior


      // Datos
      $msg = array('mensaje' => utf8_encode($msg));
```

```
        // Collapse_key: identidicador del mensaje. Como debe ser único
        para que lleguen todos los mensajes  y que no se acumulen  hago
        uso de la fecha en milisegundos
        $collapseKey = (string)microtime();

        //Generacion de los datos para crear json: mensaje, id y lista de
        moviles a la que se enviará
        $data = array(
            'data' => $msg,
            'collapse_key' => $collapseKey,
            'registration_ids' => $listMobiles
        );

        // Petición
        $ch = curl_init();

        //Configuracion de la cabecera http
        $header          =          array('Content-Type:application/json',
        Authorization:key=$apiKey");

        curl_setopt( $ch, CURLOPT_HTTPHEADER, $header);

        //Direccion url a capturar
        curl_setopt(                    $ch,                    CURLOPT_URL,
        "https://android.googleapis.com/gcm/send");

        //Comprobar que existe un nombre común en el peer: con 0 se salta
        la comprobación
        curl_setopt( $ch, CURLOPT_SSL_VERIFYHOST, 0);
        curl_setopt( $ch, CURLOPT_SSL_VERIFYPEER, 0);

        //TRUE para devolver el resultado de la transferencia como string
        del valor de curl_exec()
        curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true);

        //Todos los datos para enviar vía HTTP "POST"
        curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode($data));

        // Conectamos y recuperamos la respuesta
        $response = curl_exec($ch);

        echo("<br>La respuesta del servidor es : ".$response."<br>");
        // Cierra la conexión
        curl_close($ch);
}
```

3. En la aplicación Android crear una actividad para registrar el dispositivo en el servicio GCM y obtener un identificador.

```
import com.google.android.gms.gcm.GoogleCloudMessaging;
...

public class RegisterGCM extends Activity{

        String TAG = "Debug_RegisterGCM";

        RegisterGCM regGCM = new RegisterGCM();
        String SENDERIDGCM = ...; //Número obtenido en el paso 1
```

```
    public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          ...
          RegisterGCM regGCM = new RegisterGCM();
          regGCM.execute();
    }

    /**
     * Clase RegisterGCM: crea nueva id GCM.
     * @author Miguel Rojo Esteva
     */
    private class RegisterGCM extends
    AsyncTask<String,Integer,String>
    {
          private ProgressDialog waitDialog;
          /**
          * Muestra un mensaje mientras se registra el dispositivo
          * @see android.os.AsyncTask#onPreExecute()
          */
          protected void onPreExecute() {
                waitDialog = ProgressDialog.show(RegisterGCM.this,
                "Por favor, espere", "Registrando el dispositivo",
                true, true);
                waitDialog.setCancelable(false);
          }

          /**
           * Comprueba si el procedimiento ha finalizado con exito
           * @see android.os.AsyncTask#onPreExecute()
           */
          protected void onPostExecute(Boolean result) {

                waitDialog.dismiss();
                ...
          }

          @Override
          protected String doInBackground(String... params)
          {
                try
                {
                   GoogleCloudMessaging gcm =
                   GoogleCloudMessaging.getInstance(RegisterGCM.this)

                   //Registro en servidores de GCM
                   String id = gcm.register(SENDERIDGCM);

                   //En id está la clave que hay que utilizar para
                   enviar mensajes a ese dispositivo (la que hay que
                   establecer en $listMobiles).

                      .... Operaciones que se quieran realizar

                }catch (IOException ex){
                   Log.d(TAG, "Error registro en GCM:" +
                   ex.getMessage());
                   }
                   return "";
          }
     }
}
```

4. Para recibir los mensajes GCM hay que crear un servicio que lo reciba por broadcast.

   - Editar el fichero de manifiesto:

```
<receiver
   android:name=".GCMBroadcastReceiver"
   android:permission="com.google.android.c2dm.permission.SEND" >
   <intent-filter>
      <action android:name="com.google.android.c2dm.intent.RECEIVE" />
      <action
      android:name="com.google.android.c2dm.intent.REGISTRATION" />
      <category android:name="..." />
   </intent-filter>
</receiver>

<service android:name=".GCMIntentService" />
```

   - Crear la clase que recibirá los mensajes GCMBroadcastReceiver.

```
public class GCMBroadcastReceiver extends WakefulBroadcastReceiver
{
      @Override
      public void onReceive(Context context, Intent intent)
      {
            //Al recibir el mensaje push llama  a GCMIntentService
            ComponentName comp = new
            ComponentName(context.getPackageName(),
                        GCMIntentService.class.getName());

            //Inicia el servicio para que realice el trabajo: este
            proceso mantiene el telefono activo (wake lock)
            startWakefulService(context, (intent.setComponent(comp)));

            setResultCode(Activity.RESULT_OK);
      }
}
```

   - Servicio que ejecuta el mensaje (GCMIntentService):

```
public class GCMIntentService extends Service {

      @Override
      public IBinder onBind(Intent arg0) {
            // TODO Auto-generated method stub
            return null;
      }


      /**
       * El servicio se ejecuta hasta que es parado explicitamente,
       * por ello devuelve START_STICKY
       */
      @Override
      public int onStartCommand(Intent intent, int flags, int startId)
      {
```

```
            if(intent !=null){

                  //Obtiene el mensaje push enviado por GCM y extrae el
                  json
                  GoogleCloudMessaging gcm =
                  GoogleCloudMessaging.getInstance(this);

                  String messageType = gcm.getMessageType(intent);
                  Bundle extras = intent.getExtras();

                  //Comprueba si el mensaje es correcto (no vacio)
                  if (!extras.isEmpty()){
                        if
                        (GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE.equa
                        ls(messageType))
                        {
                              //El texto del campo mensaje del json que
                              ha llegado por GCM se encuentra en
                              extras.getString("mensaje");

                              ...
                        }
                  }
            }
            return START_STICKY;

      }
}
```

5. Añadir Google Play Services al proyecto. Para ello es necesario seguir los pasos tener indicados en http://developer.android.com/google/play-services/setup.html.